龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

龙芯 3A4000/3B4000 处理器

寄存器使用手册

多核处理器架构、寄存器描述与系统软件编程指南

V1.7

龙芯中科技术股份有限公司

北京市海淀区温泉镇中关村环保科技示范园龙芯产业园2号楼 100095
Loongson Industrial Park, building 2，Zhongguancun environmental protection park
Haidian District，Beijing

www.loongson.cn

版权声明

免责声明

本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

龙芯中科技术股份有限公司

Loongson Technology Corporation Limited
地址：北京市海淀区中关村环保科技示范园龙芯产业园 2 号楼
Building No.2, Loongson Industrial Park,
Zhongguancun Environmental Protection Park, Haidian District, Beijing
电话(Tel)：010-62546668
传真(Fax)：010-62600826

阅读指南

《龙芯 3A4000/3B4000 处理器寄存器使用手册》介绍龙芯 3A4000/3B4000 多核处理器架构与寄存器描述，对芯片系统架构、主要模块的功能与配置、寄存器列表及位域进行详细说明。

修订历史

文档更新记录	文档名:	龙芯 3A4000/3B4000 处理器寄存器使用手册
	版本号	V1.7
	创建人	芯片研发部
	创建日期	2021-07-28

更新历史

序号	更新日期	版本号	更新内容
1	2018-05-08	V0.1	初始版本
2	2018-05-28	V0.2	更新各种芯片配置寄存器
3	2018-06-02	V0.3	增加分频控制章节，更新 GPIO、UART、I2C、SPI 内容
4	2018-06-04	V0.4	修改路由
5	2018-06-05	V0.5	修改内存控制器章节，增加软件时钟系统
6	2018-06-13	V0.6	增加时钟描述；增加 GPIO 中断描述；增加温度状态检测；增加 HT 中断说明；增加 3A3000 兼容说明；修改 EXTIOI，支持 8 结点互连；修改处理器核描述。
7	2018-09-11	V0.7	更新配置寄存器列表，增加部分寄存器字段描述。
8	2018-09-13	V0.8	更新 stable clock 结构描述。
9	2018-10-26	V0.9	更新 DDR 部分
10	2019-02-19	V1.0	内部调试版本
11	2019-05-29	V1.1	更新配置寄存器、温度传感器、stable counter、扩展中断、Scache 中断、处理器 CPUCFG 部分描述
12	2019-07-01	V1.2	修改了部分语句错误
13	2019-09-11	V1.3	第 4 章增加部分特性控制说明 11.1.1 修正 int_edge 地址偏移
14	2019-10-11	V1.4	12.5 修正温度传感器寄存器描述
15	2019-12-17	V1.5	修正部分格式错误
16	2020-07-16	V1.6	增加温度中断寄存器高位描述 4.6 节增加引脚驱动位域说明

			增加部分保留位域说明
17	2021-07-28	V1.7	修订龙芯系列处理器介绍

手册信息反馈: service@loongson.cn

也可通过问题反馈网站 http://bugs.loongnix.org/ 向我司提交芯片产品使用过程中的问题，并获取技术支持。

目 录

图 目 录

表 目 录

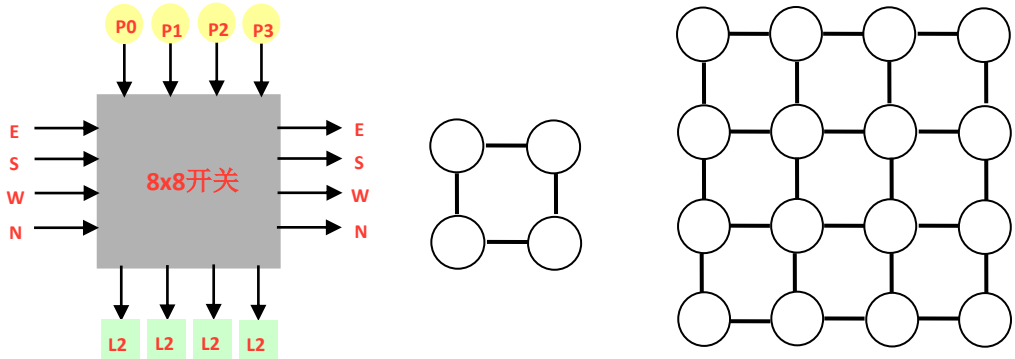
1 概述

1.1 龙芯系列处理器介绍

龙芯处理器主要包括三个系列。龙芯 1 号系列处理器采用 32 位处理器核，集成各种外围接口，形成面向特定应用的单片解决方案，主要应用于物联终端、仪器设备、数据采集等领域。龙芯 2 号系列处理器采用 32 位或 64 位处理器核，集成各种外围接口，形成面向网络设备、行业终端、智能制造等的高性能低功耗 SoC 芯片。龙芯 3 号系列处理器片内集成多个 64 位处理器核以及必要的存储和 IO 接口，面向高端嵌入式计算机、桌面、服务器等应用。

龙芯 3 号多核系列处理器基于可伸缩的多核互连架构设计，在单个芯片上集成多个高性能处理器核以及大量的 2 级 Cache，还通过高速 I/O 接口实现多芯片的互连以组成更大规模的系统。

龙芯 3 号采用的可伸缩互连结构如下图 1-1 所示。龙芯 3 号片内及多片系统通过类似的互连端口以结点为单位实现互连结构，其中每个结点由 8*8 的交叉开关组成，每个交叉开关连接四个处理器核以及四个共享 Cache，并与东（E）南（N）西（W）北（N）四个方向的其他结点互连。



龙芯 3 号结点及二维互连结构，（a）结点结构，（b）2*2 的 mesh 网络连接 16 个处理器，（c）4*4 的 mesh 网络连接 64 个处理器。

图 1-1 龙芯 3 号系统结构

龙芯 3 号的结点结构如下图 1-2 所示。每个结点有两级 AXI 交叉开关连接处理器、共

享 Cache、内存控制器以及 IO 控制器。其中第一级 AXI 交叉开关（称为 X1 Switch，简称 X1）连接处理器和共享 Cache。第二级交叉开关（称为 X2 Switch，简称 X2）连接共享 Cache 和内存控制器。

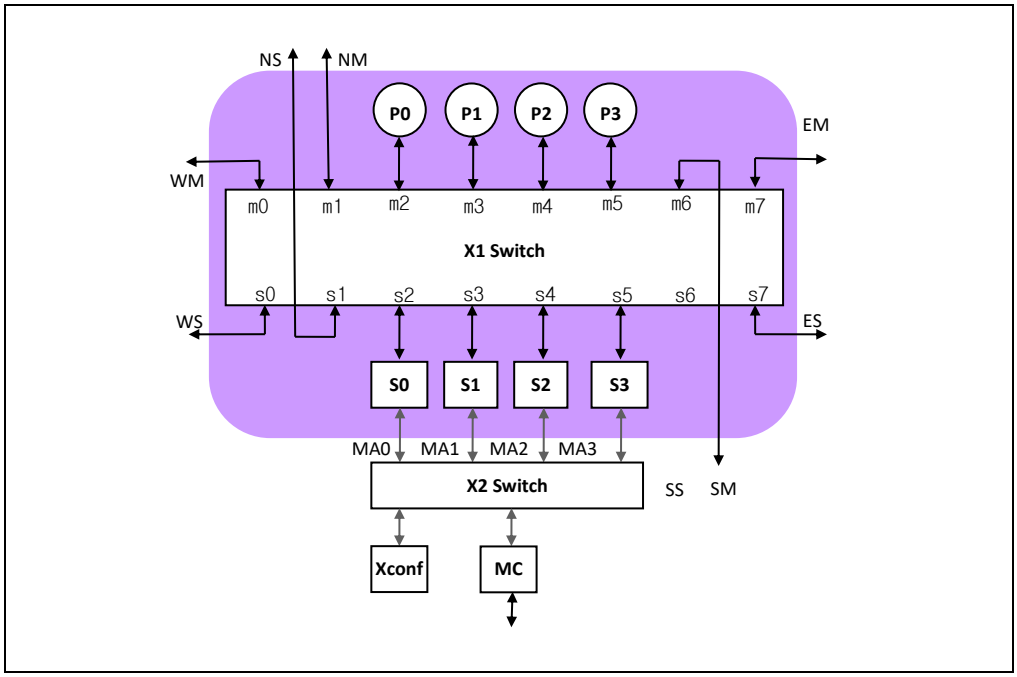


图 1-2 龙芯 3 号结点结构

在每个结点中，最多 8*8 的 X1 交叉开关通过四个 Master 端口连接四个 GS464 处理器核（图中 P0、P1、P2、P3），通过四个 Slave 端口连接统一编址的四个 interleave 共享 Cache 块（图中 S0、S1、S2、S3），通过四对 Master/Slave 端口连接东、南、西、北四个方向的其他结点或 IO 结点（图中 EM/ES、SM/SS、WM/WS、NM/NS）。

X2 交叉开关通过四个 Master 端口连接四个共享 Cache，至少一个 Slave 端口连接一个内存控制器，至少一个 Slave 端口连接一个交叉开关的配置模块（Xconf），该配置模块用于配置本结点的 X1 和 X2 的地址窗口等。还可以根据需要连接更多的内存控制器和 IO 端口等。

1.2 龙芯 3A4000/3B4000 简介

龙芯 3A4000 是一款四核龙芯处理器，采用 28nm 工艺制造，稳定工作频率为 1.5-2.0GHz，主要技术特征如下：

- 片内集成 4 个 64 位的四发射超标量 GS464V 高性能处理器核；
- 片内集成 8MB 的分体共享三级 Cache（由 4 个体模块组成，每个体模块容量为 2MB）；
- 通过目录协议维护多核及 I/O DMA 访问的 Cache 一致性；

- 片内集成 2 个 72 位带 ECC 的 DDR4 控制器；
- 片内集成 2 个 16 位的 HyperTransport 控制器（以下简称 HT）；
- 每个 16 位的 HT 端口拆分成两个 8 路的 HT 端口使用；
- 片内集成 2 个 I2C、1 个 UART、1 个 SPI、16 路 GPIO 接口。

龙芯 3A4000 的顶层结构设计在 3A2000/3A3000 基础上进行较大幅度的优化，其主要改进如下：

- 调整了片上互连结构，简化了地址路由，IO 模块间互连采用 RING 结构；
- 优化了 HT 控制器的带宽利用率与跨片延时；
- 优化了内存控制器结构，增加了内存控制器 DDR4 的支持，并支持内存槽连接加速卡；
- 规范了配置寄存器空间与访问方式，引入了 CSR 配置寄存器访问机制；
- 优化了中断控制器结构，支持向量中断硬件分发机制；

龙芯 3B4000 与龙芯 3A4000 相比，可以支持多路互连。龙芯 3A4000 的部分版本不支持多路互连。其它的软件寄存器配置基本一致。下文不再单独区分 3A4000 和 3B4000，统一以 3A4000 指代。

龙芯 3A4000 芯片整体架构基于多级互连实现，结构如下图 1-3 所示。

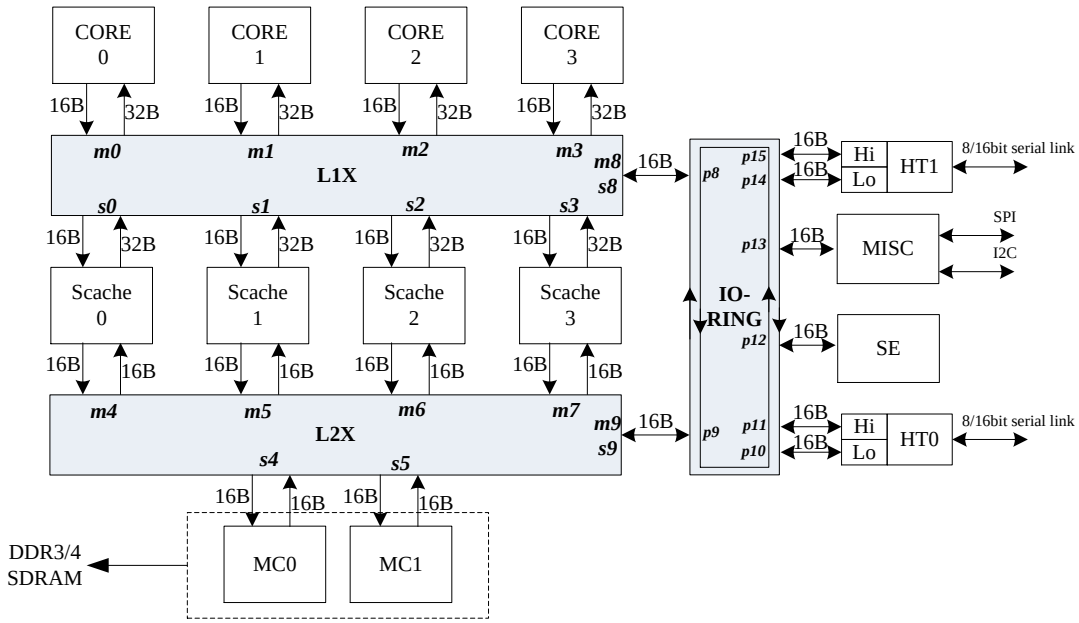


图 1-3 龙芯 3A4000 芯片结构

第一级互连采用 5x5 的交叉开关，用于连接四个 GS464v 核（作为主设备）、四个共享 Cache 模块（作为从设备）、以及一个 IO 端口连接 IO-RING。（IO 端口使用一个 Master 和

一个 Slave）。

第二级互连采用 5x3 的交叉开关，连接 4 个共享 Cache 模块（作为主设备），两个 DDR3/4 内存控制器、以及一个 IO 端口连接 IO-RING。

IO-RING 包含 8 个端口，连接包括 4 个 HT 控制器，MISC 模块，SE 模块与两级交叉开关。两个 HT 控制器（lo/hi）共用 16 位 HT 总线，作为两个 8 位的 HT 总线使用，也可以由 lo 独占 16 位 HT 总线。HT 控制器内集成一个 DMA 控制器，DMA 控制器负责 IO 的 DMA 控制并负责片间一致性的维护。

上述互连结构都采用读写分离的数据通道，数据通道宽度为 128bit，工作在与处理器核相同的频率，用以提供高速的片上数据传输。此外，一级交叉开关连接 4 个处理器核与 scache 的读数据通道为 256 位，以提高片内处理器核访问 scache 的读带宽。

2 系统配置与控制

2.1 芯片工作模式

根据组成系统的结构，龙芯 3A4000 主要包括两种工作模式：

- 单芯片模式。系统只包含 1 片龙芯 3A4000，是一个对称多处理器系统（SMP）；
- 多芯片互连模式。系统中包含 2 片、4 片或 8 片龙芯 3A4000，通过 HT 端口进行互连，构成一个非均匀访存多处理器系统（CC-NUMA）。

2.2 控制引脚说明

主要控制引脚包括 DO_TEST、ICCC_EN、NODE_ID[2:0]、CLKSEL[9:0]、CHIP_CONFIG[5:0]。

表 2-1 控制引脚说明

<table>
<tr><th>信号</th><th>上下拉</th><th>作用</th></tr>
<tr><td>DO_TEST</td><td>上拉</td><td>1’b1 表示功能模式
1’b0 表示测试模式</td></tr>
<tr><td>ICCC_EN</td><td>下拉</td><td>1’b1 表示多芯片一致性互连模式
1’b0 表示单芯片模式</td></tr>
<tr><td>NODE_ID[2:0]</td><td></td><td>在多芯片一致性互连模式下表示处理器号</td></tr>
<tr><td>CLKSEL[9:0]</td><td></td><td>HT 时钟控制
<table>
<tr><th>信号</th><th>作用</th></tr>
<tr><td>CLKSEL[9]</td><td>1’ b1：表示 HT PLL 时钟采用 CLKSEL[9:4]控制
1’ b0：初始倍频为 1 倍频，可由软件进行重新配置</td></tr>
<tr><td>CLKSEL[8]</td><td>1’ b1：表示 HT PLL 采用 SYSCLK 时钟输入
1’ b0：表示 HT PLL 采用差分时钟输入</td></tr>
<tr><td>CLKSEL[7:6]</td><td>2’ b00 表示 PHY 时钟频率为 1.6GHz
2’ b01 表示 PHY 时钟频率为 3.2GHz（参考时钟为 25MHz 时为 1.6GHz）
2’ b10 表示 PHY 时钟频率为 1.2GHz
2’ b11 表示 PHY 时钟频率为 2.4GHz</td></tr>
<tr><td>CLKSEL[5]</td><td>保留</td></tr>
<tr><td>CLKSEL[4]</td><td>1-参考时钟采用 25MHz，0-参考时钟采用 100MHz</td></tr>
</table></td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>
MEM 时钟控制（时钟频率应为接口时钟 1/2）
<table>
<tr><th>CLKSEL[3:2]</th><th>输出频率</th></tr>
<tr><td>2’b00</td><td>466MHz</td></tr>
<tr><td>2’b01</td><td>600MHz</td></tr>
<tr><td>2’b10</td><td>软件配置（PLL 时钟倍频 1.6-3.2GHz）</td></tr>
<tr><td>2’b11</td><td>SYSCLK（100MHz/25MHz）</td></tr>
</table>
Main 时钟控制（网络与处理器核最高频率）
<table>
<tr><th>CLKSEL[1:0]</th><th>输出频率</th></tr>
<tr><td>2’b00</td><td>1GHz</td></tr>
<tr><td>2’b01</td><td>2GHz</td></tr>
<tr><td>2’b10</td><td>软件配置（PLL 时钟倍频 1.6-3.2GHz）</td></tr>
<tr><td>2’b11</td><td>SYSCLK（100MHz/25MHz）</td></tr>
</table>
</td>
</tr>
<tr>
<td>CHIP_CONFIG[5:0]</td>
<td></td>
<td>
芯片配置控制
<table>
<tr><td>CHIP_CONFIG[0]</td><td>SE 功能使能</td></tr>
<tr><td>CHIP_CONFIG[1]</td><td>默认 HT Gen1 模式</td></tr>
<tr><td>CHIP_CONFIG[2]</td><td>保留</td></tr>
<tr><td>CHIP_CONFIG[3]</td><td>HT0/1-hi 默认进入一致性模式，用于支持 8 路互连</td></tr>
<tr><td>CHIP_CONFIG[4]</td><td>HT 逻辑功能互换，HT0/HT1 交换</td></tr>
<tr><td>CHIP_CONFIG[5]</td><td>片内时钟调试使能（DCDL）</td></tr>
</table>
</td>
</tr>
</table>

3 物理地址空间分布

龙芯 3 号系列处理器的系统物理地址分布采用全局可访问的层次化寻址设计，以保证系统开发的扩展兼容。整个系统的物理地址宽度为 48 位。按照地址的高 4 位，整个地址空间被均匀分布到 16 个结点上，即每个结点分配 44 位地址空间。

3.1 结点间的物理地址空间分布

龙芯 3A4000 处理器可以直接采用 2/4/8 片 3A4000 芯片直连构建 CC-NUMA 系统，每个芯片的处理器号由引脚 NODEID 决定，每个芯片的地址空间分布如下：

表 3-1 结点级的系统全局地址分布

芯片结点号（NODEID）	地址[47:44]位	起始地址	结束地址
0	0	0x0000_0000_0000	0x0FFF_FFFF_FFFF
1	1	0x1000_0000_0000	0x1FFF_FFFF_FFFF
2	2	0x2000_0000_0000	0x2FFF_FFFF_FFFF
3	3	0x3000_0000_0000	0x3FFF_FFFF_FFFF
4	4	0x4000_0000_0000	0x4FFF_FFFF_FFFF
5	5	0x5000_0000_0000	0x5FFF_FFFF_FFFF
6	6	0x6000_0000_0000	0x6FFF_FFFF_FFFF
7	7	0x7000_0000_0000	0x7FFF_FFFF_FFFF

当系统结点数目不足8个结点时，应设置路由设置寄存器(0x1fe00400)的nodemask字段，当发生猜测访问时，保证即使没有物理结点的地址也能够得到响应。（2路：0x1;4路：0x3）

3.2 结点内的物理地址空间分布

龙芯 3A4000 采用单结点 4 核配置，因此龙芯 3A4000 芯片集成的 DDR 内存控制器、HT 总线的对应地址都包含在从 0x0（含）至 0x1000_0000_0000（不含）的 44 位地址空间内。在每个结点的内部，44 位地址空间又进一步划分给结点内连接的所有设备，仅当访问类型为 cached 时，请求会被路由到 4 个共享 Cache 模块。根据芯片和系统结构配置的不同，如果某端口上没有连接从设备，则对应的地址空间是保留地址空间，不允许访问。

龙芯 3A4000 芯片内部互连的地址

空间对应的各个从设备端如下：

表 3-2 结点内的地址分布

设备	地址[43:40]位	结点内起始地址	结点结束地址
MC0	4	0x400_0000_0000	0x4FF_FFFF_FFFF
MC1	5	0x500_0000_0000	0x5FF_FFFF_FFFF
SE	c	0xC00_0000_0000	0xCFF_FFFF_FFFF
HT0 Lo 控制器	a	0xA00_0000_0000	0xAFF_FFFF_FFFF
HT0 Hi 控制器	b	0xB00_0000_0000	0xBFF_FFFF_FFFF
HT1 Lo 控制器	e	0xE00_0000_0000	0xEFF_FFFF_FFFF
HT1 Hi 控制器	f	0xF00_0000_0000	0xFFF_FFFF_FFFF

不同于方向端口的映射关系，龙芯 3A4000 可以根据实际应用的访问行为，来决定共享 Cache 的交叉寻址方式。结点内的 4 个共享 Cache 模块所对应的地址空间根据地址位的某两位选择位确定，并可以通过软件进行动态配置修改。系统中设置了名为 SCID_SEL 的配置寄存器来确定地址选择位，如下表所示。在缺省情况下采用[7:6]地址散列的方式进行分布，即地址[7:6]两位决定对应的共享 Cache 编号。该寄存器地址 0x3FF00400 或者 0x1fe00400。

表 3-3 SCID_SEL 地址位设置

SCID_SEL	地址位选择	SCID_SEL	地址位选择
4'h0	7: 6	4'h8	23:22
4'h1	9: 8	4'h9	25:24
4'h2	11:10	4'ha	27:26
4'h3	13:12	4'hb	29:28
4'h4	15:14	4'hc	31:30
4'h5	17:16	4'hd	33:32
4'h6	19:18	4'he	35:34
4'h7	21:20	4'hf	37:36

龙芯 3A4000 处理器每个结点的内部 44 位物理地址的默认分布如下表所示：

表 3-4 结点内 44 位物理地址分布

地址范围	访问属性	目的地
addr[43:40]==4’ha	本地结点,uncache	HT0_LO
addr[43:40]==4’hb	本地结点,uncache	HT0_HI
addr[43:40]==4’hc	本地结点,uncache	SE

addr[43:40]==4'he	本地结点,uncache	HT1_LO
addr[43:40]==4'hf	本地结点,uncache	HT1_HI
0x10000000-0x1fffffff, 0x3ff00000-0x3ff0ffff(可关闭)	本地结点,uncache	MISC
Mc interleave 为 0，且非以上地址	本地结点,uncache	MC0
Mc interleave 为 1，且非以上地址	本地结点,uncache	MC1
Scache interleave 为 0(由 scid_sel 决定的地址位选择)	本地结点,cache	Scache0
Scache interleave 为 1(由 scid_sel 决定的地址位选择)	本地结点,cache	Scache1
Scache interleave 为 2(由 scid_sel 决定的地址位选择)	本地结点,cache	Scache2
Scache interleave 为 3(由 scid_sel 决定的地址位选择)	本地结点,cache	Scache3

3.3 地址路由分布与配置

龙芯 3A4000 的路由主要通过系统的两级交叉开关与 IO-RING 实现。软件可以对每个 Master 端口接收到的请求进行路由配置，每个 Master 端口都拥有 8 个地址窗口，可以完成 8 个地址窗口的目标路由选择。每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 K 字节对齐；MASK 采用类似网络掩码高位为 1 的格式；MMAP 的低四位表示对应目标 Slave 端口的编号，MMAP[4]表示允许取指，MMAP[5]表示允许块读，MMAP[6]表示允许交错访问使能，MMAP[7]表示窗口使能。

表 3-5 MMAP 字段对应的该空间访问属性

[7]	[6]	[5]	[4]
窗口使能	允许对 SCACHE/内存进行交错访问	允许块读	允许取指

窗口命中公式:　　(IN_ADDR & MASK) == BASE

由于龙芯 3 号缺省采用固定路由，在上电启动时，配置窗口都处于关闭状态，使用时需要系统软件对其进行使能配置。

SCACHE/内存交错访问配置使能后，Slave 号仅为 0 或 4 时有效。为 0 表示路由到 SCACHE，并由 SCID_SEL 决定如何在 4 个 SCACHE 进行交错访问。为 4 表示路由到内存，由 interleave_bit 决定如何在 2 个 MC 进行交错访问。

地址窗口转换寄存器如下表所示。

表 3-6 地址窗口寄存器表

地址	寄存器	地址	寄存器
0x3ff0_2000	CORE0_WIN0_BASE	0x3ff0_2100	CORE1_WIN0_BASE
0x3ff0_2008	CORE0_WIN1_BASE	0x3ff0_2108	CORE1_WIN1_BASE
0x3ff0_2010	CORE0_WIN2_BASE	0x3ff0_2110	CORE1_WIN2_BASE
0x3ff0_2018	CORE0_WIN3_BASE	0x3ff0_2118	CORE1_WIN3_BASE
0x3ff0_2020	CORE0_WIN4_BASE	0x3ff0_2120	CORE1_WIN4_BASE

0x3ff0_2028	CORE0_WIN5_BASE	0x3ff0_2128	CORE1_WIN5_BASE
0x3ff0_2030	CORE0_WIN6_BASE	0x3ff0_2130	CORE1_WIN6_BASE
0x3ff0_2038	CORE0_WIN7_BASE	0x3ff0_2138	CORE1_WIN7_BASE
0x3ff0_2040	CORE0_WIN0_MASK	0x3ff0_2140	CORE1_WIN0_MASK
0x3ff0_2048	CORE0_WIN1_MASK	0x3ff0_2148	CORE1_WIN1_MASK
0x3ff0_2050	CORE0_WIN2_MASK	0x3ff0_2150	CORE1_WIN2_MASK
0x3ff0_2058	CORE0_WIN3_MASK	0x3ff0_2158	CORE1_WIN3_MASK
0x3ff0_2060	CORE0_WIN4_MASK	0x3ff0_2160	CORE1_WIN4_MASK
0x3ff0_2068	CORE0_WIN5_MASK	0x3ff0_2168	CORE1_WIN5_MASK
0x3ff0_2070	CORE0_WIN6_MASK	0x3ff0_2170	CORE1_WIN6_MASK
0x3ff0_2078	CORE0_WIN7_MASK	0x3ff0_2178	CORE1_WIN7_MASK
0x3ff0_2080	CORE0_WIN0_MMAP	0x3ff0_2180	CORE1_WIN0_MMAP
0x3ff0_2088	CORE0_WIN1_MMAP	0x3ff0_2188	CORE1_WIN1_MMAP
0x3ff0_2090	CORE0_WIN2_MMAP	0x3ff0_2190	CORE1_WIN2_MMAP
0x3ff0_2098	CORE0_WIN3_MMAP	0x3ff0_2198	CORE1_WIN3_MMAP
0x3ff0_20a0	CORE0_WIN4_MMAP	0x3ff0_21a0	CORE1_WIN4_MMAP
0x3ff0_20a8	CORE0_WIN5_MMAP	0x3ff0_21a8	CORE1_WIN5_MMAP
0x3ff0_20b0	CORE0_WIN6_MMAP	0x3ff0_21b0	CORE1_WIN6_MMAP
0x3ff0_20b8	CORE0_WIN7_MMAP	0x3ff0_21b8	CORE1_WIN7_MMAP
0x3ff0_2200	CORE2_WIN0_BASE	0x3ff0_2300	CORE3_WIN0_BASE
0x3ff0_2208	CORE2_WIN1_BASE	0x3ff0_2308	CORE3_WIN1_BASE
0x3ff0_2210	CORE2_WIN2_BASE	0x3ff0_2310	CORE3_WIN2_BASE
0x3ff0_2218	CORE2_WIN3_BASE	0x3ff0_2318	CORE3_WIN3_BASE
0x3ff0_2220	CORE2_WIN4_BASE	0x3ff0_2320	CORE3_WIN4_BASE
0x3ff0_2228	CORE2_WIN5_BASE	0x3ff0_2328	CORE3_WIN5_BASE
0x3ff0_2230	CORE2_WIN6_BASE	0x3ff0_2330	CORE3_WIN6_BASE
0x3ff0_2238	CORE2_WIN7_BASE	0x3ff0_2338	CORE3_WIN7_BASE
0x3ff0_2240	CORE2_WIN0_MASK	0x3ff0_2340	CORE3_WIN0_MASK
0x3ff0_2248	CORE2_WIN1_MASK	0x3ff0_2348	CORE3_WIN1_MASK
0x3ff0_2250	CORE2_WIN2_MASK	0x3ff0_2350	CORE3_WIN2_MASK
0x3ff0_2258	CORE2_WIN3_MASK	0x3ff0_2358	CORE3_WIN3_MASK
0x3ff0_2260	CORE2_WIN4_MASK	0x3ff0_2360	CORE3_WIN4_MASK
0x3ff0_2268	CORE2_WIN5_MASK	0x3ff0_2368	CORE3_WIN5_MASK
0x3ff0_2270	CORE2_WIN6_MASK	0x3ff0_2370	CORE3_WIN6_MASK
0x3ff0_2278	CORE2_WIN7_MASK	0x3ff0_2378	CORE3_WIN7_MASK
0x3ff0_2280	CORE2_WIN0_MMAP	0x3ff0_2380	CORE3_WIN0_MMAP
0x3ff0_2288	CORE2_WIN1_MMAP	0x3ff0_2388	CORE3_WIN1_MMAP
0x3ff0_2290	CORE2_WIN2_MMAP	0x3ff0_2390	CORE3_WIN2_MMAP
0x3ff0_2298	CORE2_WIN3_MMAP	0x3ff0_2398	CORE3_WIN3_MMAP
0x3ff0_22a0	CORE2_WIN4_MMAP	0x3ff0_23a0	CORE3_WIN4_MMAP
0x3ff0_22a8	CORE2_WIN5_MMAP	0x3ff0_23a8	CORE3_WIN5_MMAP

0x3ff0_22b0	CORE2_WIN6_MMAP	0x3ff0_23b0	CORE3_WIN6_MMAP
0x3ff0_22b8	CORE2_WIN7_MMAP	0x3ff0_23b8	CORE3_WIN7_MMAP
0x3ff0_2400	SCACHE0_WIN0_BASE	0x3ff0_2500	SCACHE1_WIN0_BASE
0x3ff0_2408	SCACHE0_WIN1_BASE	0x3ff0_2508	SCACHE1_WIN1_BASE
0x3ff0_2410	SCACHE0_WIN2_BASE	0x3ff0_2510	SCACHE1_WIN2_BASE
0x3ff0_2418	SCACHE0_WIN3_BASE	0x3ff0_2518	SCACHE1_WIN3_BASE
0x3ff0_2420	SCACHE0_WIN4_BASE	0x3ff0_2520	SCACHE1_WIN4_BASE
0x3ff0_2428	SCACHE0_WIN5_BASE	0x3ff0_2528	SCACHE1_WIN5_BASE
0x3ff0_2430	SCACHE0_WIN6_BASE	0x3ff0_2530	SCACHE1_WIN6_BASE
0x3ff0_2438	SCACHE0_WIN7_BASE	0x3ff0_2538	SCACHE1_WIN7_BASE
0x3ff0_2440	SCACHE0_WIN0_MASK	0x3ff0_2540	SCACHE1_WIN0_MASK
0x3ff0_2448	SCACHE0_WIN1_MASK	0x3ff0_2548	SCACHE1_WIN1_MASK
0x3ff0_2450	SCACHE0_WIN2_MASK	0x3ff0_2550	SCACHE1_WIN2_MASK
0x3ff0_2458	SCACHE0_WIN3_MASK	0x3ff0_2558	SCACHE1_WIN3_MASK
0x3ff0_2460	SCACHE0_WIN4_MASK	0x3ff0_2560	SCACHE1_WIN4_MASK
0x3ff0_2468	SCACHE0_WIN5_MASK	0x3ff0_2568	SCACHE1_WIN5_MASK
0x3ff0_2470	SCACHE0_WIN6_MASK	0x3ff0_2570	SCACHE1_WIN6_MASK
0x3ff0_2478	SCACHE0_WIN7_MASK	0x3ff0_2578	SCACHE1_WIN7_MASK
0x3ff0_2480	SCACHE0_WIN0_MMAP	0x3ff0_2580	SCACHE1_WIN0_MMAP
0x3ff0_2488	SCACHE0_WIN1_MMAP	0x3ff0_2588	SCACHE1_WIN1_MMAP
0x3ff0_2490	SCACHE0_WIN2_MMAP	0x3ff0_2590	SCACHE1_WIN2_MMAP
0x3ff0_2498	SCACHE0_WIN3_MMAP	0x3ff0_2598	SCACHE1_WIN3_MMAP
0x3ff0_24a0	SCACHE0_WIN4_MMAP	0x3ff0_25a0	SCACHE1_WIN4_MMAP
0x3ff0_24a8	SCACHE0_WIN5_MMAP	0x3ff0_25a8	SCACHE1_WIN5_MMAP
0x3ff0_24b0	SCACHE0_WIN6_MMAP	0x3ff0_25b0	SCACHE1_WIN6_MMAP
0x3ff0_24b8	SCACHE0_WIN7_MMAP	0x3ff0_25b8	SCACHE1_WIN7_MMAP
0x3ff0_2600	SCACHE2_WIN0_BASE	0x3ff0_2700	SCACHE3_WIN0_BASE
0x3ff0_2608	SCACHE2_WIN1_BASE	0x3ff0_2708	SCACHE3_WIN1_BASE
0x3ff0_2610	SCACHE2_WIN2_BASE	0x3ff0_2710	SCACHE3_WIN2_BASE
0x3ff0_2618	SCACHE2_WIN3_BASE	0x3ff0_2718	SCACHE3_WIN3_BASE
0x3ff0_2620	SCACHE2_WIN4_BASE	0x3ff0_2720	SCACHE3_WIN4_BASE
0x3ff0_2628	SCACHE2_WIN5_BASE	0x3ff0_2728	SCACHE3_WIN5_BASE
0x3ff0_2630	SCACHE2_WIN6_BASE	0x3ff0_2730	SCACHE3_WIN6_BASE
0x3ff0_2638	SCACHE2_WIN7_BASE	0x3ff0_2738	SCACHE3_WIN7_BASE
0x3ff0_2640	SCACHE2_WIN0_MASK	0x3ff0_2740	SCACHE3_WIN0_MASK
0x3ff0_2648	SCACHE2_WIN1_MASK	0x3ff0_2748	SCACHE3_WIN1_MASK
0x3ff0_2650	SCACHE2_WIN2_MASK	0x3ff0_2750	SCACHE3_WIN2_MASK
0x3ff0_2658	SCACHE2_WIN3_MASK	0x3ff0_2758	SCACHE3_WIN3_MASK
0x3ff0_2660	SCACHE2_WIN4_MASK	0x3ff0_2760	SCACHE3_WIN4_MASK
0x3ff0_2668	SCACHE2_WIN5_MASK	0x3ff0_2768	SCACHE3_WIN5_MASK

0x3ff0_2670	SCACHE2_WIN6_MASK	0x3ff0_2770	SCACHE3_WIN6_MASK
0x3ff0_2678	SCACHE2_WIN7_MASK	0x3ff0_2778	SCACHE3_WIN7_MASK
0x3ff0_2680	SCACHE2_WIN0_MMAP	0x3ff0_2780	SCACHE3_WIN0_MMAP
0x3ff0_2688	SCACHE2_WIN1_MMAP	0x3ff0_2788	SCACHE3_WIN1_MMAP
0x3ff0_2690	SCACHE2_WIN2_MMAP	0x3ff0_2790	SCACHE3_WIN2_MMAP
0x3ff0_2698	SCACHE2_WIN3_MMAP	0x3ff0_2798	SCACHE3_WIN3_MMAP
0x3ff0_26a0	SCACHE2_WIN4_MMAP	0x3ff0_27a0	SCACHE3_WIN4_MMAP
0x3ff0_26a8	SCACHE2_WIN5_MMAP	0x3ff0_27a8	SCACHE3_WIN5_MMAP
0x3ff0_26b0	SCACHE2_WIN6_MMAP	0x3ff0_27b0	SCACHE3_WIN6_MMAP
0x3ff0_26b8	SCACHE2_WIN7_MMAP	0x3ff0_27b8	SCACHE3_WIN7_MMAP
-	-	0x3ff0_2900	IO_L2X_WIN0_BASE
-	-	0x3ff0_2908	IO_L2X_WIN1_BASE
-	-	0x3ff0_2910	IO_L2X_WIN2_BASE
-	-	0x3ff0_2918	IO_L2X_WIN3_BASE
-	-	0x3ff0_2920	IO_L2X_WIN4_BASE
-	-	0x3ff0_2928	IO_L2X_WIN5_BASE
-	-	0x3ff0_2930	IO_L2X_WIN6_BASE
-	-	0x3ff0_2938	IO_L2X_WIN7_BASE
-	-	0x3ff0_2940	IO_L2X_WIN0_MASK
-	-	0x3ff0_2948	IO_L2X_WIN1_MASK
-	-	0x3ff0_2950	IO_L2X_WIN2_MASK
-	-	0x3ff0_2958	IO_L2X_WIN3_MASK
-	-	0x3ff0_2960	IO_L2X_WIN4_MASK
-	-	0x3ff0_2968	IO_L2X_WIN5_MASK
-	-	0x3ff0_2970	IO_L2X_WIN6_MASK
-	-	0x3ff0_2978	IO_L2X_WIN7_MASK
-	-	0x3ff0_2980	IO_L2X_WIN0_MMAP
-	-	0x3ff0_2988	IO_L2X_WIN1_MMAP
-	-	0x3ff0_2990	IO_L2X_WIN2_MMAP
-	-	0x3ff0_2998	IO_L2X_WIN3_MMAP
-	-	0x3ff0_29a0	IO_L2X_WIN4_MMAP
-	-	0x3ff0_29a8	IO_L2X_WIN5_MMAP
-	-	0x3ff0_29b0	IO_L2X_WIN6_MMAP
-	-	0x3ff0_29b8	IO_L2X_WIN7_MMAP
0x3ff0_2a00	HT0_LO_WIN0_BASE	0x3ff0_2b00	HT0_HI_WIN0_BASE
0x3ff0_2a08	HT0_LO_WIN1_BASE	0x3ff0_2b08	HT0_HI_WIN1_BASE
0x3ff0_2a10	HT0_LO_WIN2_BASE	0x3ff0_2b10	HT0_HI_WIN2_BASE
0x3ff0_2a18	HT0_LO_WIN3_BASE	0x3ff0_2b18	HT0_HI_WIN3_BASE
0x3ff0_2a20	HT0_LO_WIN4_BASE	0x3ff0_2b20	HT0_HI_WIN4_BASE
0x3ff0_2a28	HT0_LO_WIN5_BASE	0x3ff0_2b28	HT0_HI_WIN5_BASE

0x3ff0_2a30	HT0_LO_WIN6_BASE	0x3ff0_2b30	HT0_HI_WIN6_BASE
0x3ff0_2a38	HT0_LO_WIN7_BASE	0x3ff0_2b38	HT0_HI_WIN7_BASE
0x3ff0_2a40	HT0_LO_WIN0_MASK	0x3ff0_2b40	HT0_HI_WIN0_MASK
0x3ff0_2a48	HT0_LO_WIN1_MASK	0x3ff0_2b48	HT0_HI_WIN1_MASK
0x3ff0_2a50	HT0_LO_WIN2_MASK	0x3ff0_2b50	HT0_HI_WIN2_MASK
0x3ff0_2a58	HT0_LO_WIN3_MASK	0x3ff0_2b58	HT0_HI_WIN3_MASK
0x3ff0_2a60	HT0_LO_WIN4_MASK	0x3ff0_2b60	HT0_HI_WIN4_MASK
0x3ff0_2a68	HT0_LO_WIN5_MASK	0x3ff0_2b68	HT0_HI_WIN5_MASK
0x3ff0_2a70	HT0_LO_WIN6_MASK	0x3ff0_2b70	HT0_HI_WIN6_MASK
0x3ff0_2a78	HT0_LO_WIN7_MASK	0x3ff0_2b78	HT0_HI_WIN7_MASK
0x3ff0_2a80	HT0_LO_WIN0_MMAP	0x3ff0_2b80	HT0_HI_WIN0_MMAP
0x3ff0_2a88	HT0_LO_WIN1_MMAP	0x3ff0_2b88	HT0_HI_WIN1_MMAP
0x3ff0_2a90	HT0_LO_WIN2_MMAP	0x3ff0_2b90	HT0_HI_WIN2_MMAP
0x3ff0_2a98	HT0_LO_WIN3_MMAP	0x3ff0_2b98	HT0_HI_WIN3_MMAP
0x3ff0_2aa0	HT0_LO_WIN4_MMAP	0x3ff0_2ba0	HT0_HI_WIN4_MMAP
0x3ff0_2aa8	HT0_LO_WIN5_MMAP	0x3ff0_2ba8	HT0_HI_WIN5_MMAP
0x3ff0_2ab0	HT0_LO_WIN6_MMAP	0x3ff0_2bb0	HT0_HI_WIN6_MMAP
0x3ff0_2ab8	HT0_LO_WIN7_MMAP	0x3ff0_2bb8	HT0_HI_WIN7_MMAP
0x3ff0_2c00	SE_WIN0_BASE	0x3ff0_2d00	MISC_WIN0_BASE
0x3ff0_2c08	SE_WIN1_BASE	0x3ff0_2d08	MISC_WIN1_BASE
0x3ff0_2c10	SE_WIN2_BASE	0x3ff0_2d10	MISC_WIN2_BASE
0x3ff0_2c18	SE_WIN3_BASE	0x3ff0_2d18	MISC_WIN3_BASE
0x3ff0_2c20	SE_WIN4_BASE	0x3ff0_2d20	MISC_WIN4_BASE
0x3ff0_2c28	SE_WIN5_BASE	0x3ff0_2d28	MISC_WIN5_BASE
0x3ff0_2c30	SE_WIN6_BASE	0x3ff0_2d30	MISC_WIN6_BASE
0x3ff0_2c38	SE_WIN7_BASE	0x3ff0_2d38	MISC_WIN7_BASE
0x3ff0_2c40	SE_WIN0_MASK	0x3ff0_2d40	MISC_WIN0_MASK
0x3ff0_2c48	SE_WIN1_MASK	0x3ff0_2d48	MISC_WIN1_MASK
0x3ff0_2c50	SE_WIN2_MASK	0x3ff0_2d50	MISC_WIN2_MASK
0x3ff0_2c58	SE_WIN3_MASK	0x3ff0_2d58	MISC_WIN3_MASK
0x3ff0_2c60	SE_WIN4_MASK	0x3ff0_2d60	MISC_WIN4_MASK
0x3ff0_2c68	SE_WIN5_MASK	0x3ff0_2d68	MISC_WIN5_MASK
0x3ff0_2c70	SE_WIN6_MASK	0x3ff0_2d70	MISC_WIN6_MASK
0x3ff0_2c78	SE_WIN7_MASK	0x3ff0_2d78	MISC_WIN7_MASK
0x3ff0_2c80	SE_WIN0_MMAP	0x3ff0_2d80	MISC_WIN0_MMAP
0x3ff0_2c88	SE_WIN1_MMAP	0x3ff0_2d88	MISC_WIN1_MMAP
0x3ff0_2c90	SE_WIN2_MMAP	0x3ff0_2d90	MISC_WIN2_MMAP
0x3ff0_2c98	SE_WIN3_MMAP	0x3ff0_2d98	MISC_WIN3_MMAP
0x3ff0_2ca0	SE_WIN4_MMAP	0x3ff0_2da0	MISC_WIN4_MMAP
0x3ff0_2ca8	SE_WIN5_MMAP	0x3ff0_2da8	MISC_WIN5_MMAP
0x3ff0_2cb0	SE_WIN6_MMAP	0x3ff0_2db0	MISC_WIN6_MMAP

0x3ff0_2cb8	SE_WIN7_MMAP	0x3ff0_2db8	MISC_WIN7_MMAP
0x3ff0_2e00	HT1_LO_WIN0_BASE	0x3ff0_2f00	HT1_HI_WIN0_BASE
0x3ff0_2e08	HT1_LO_WIN1_BASE	0x3ff0_2f08	HT1_HI_WIN1_BASE
0x3ff0_2e10	HT1_LO_WIN2_BASE	0x3ff0_2f10	HT1_HI_WIN2_BASE
0x3ff0_2e18	HT1_LO_WIN3_BASE	0x3ff0_2f18	HT1_HI_WIN3_BASE
0x3ff0_2e20	HT1_LO_WIN4_BASE	0x3ff0_2f20	HT1_HI_WIN4_BASE
0x3ff0_2e28	HT1_LO_WIN5_BASE	0x3ff0_2f28	HT1_HI_WIN5_BASE
0x3ff0_2e30	HT1_LO_WIN6_BASE	0x3ff0_2f30	HT1_HI_WIN6_BASE
0x3ff0_2e38	HT1_LO_WIN7_BASE	0x3ff0_2f38	HT1_HI_WIN7_BASE
0x3ff0_2e40	HT1_LO_WIN0_MASK	0x3ff0_2f40	HT1_HI_WIN0_MASK
0x3ff0_2e48	HT1_LO_WIN1_MASK	0x3ff0_2f48	HT1_HI_WIN1_MASK
0x3ff0_2e50	HT1_LO_WIN2_MASK	0x3ff0_2f50	HT1_HI_WIN2_MASK
0x3ff0_2e58	HT1_LO_WIN3_MASK	0x3ff0_2f58	HT1_HI_WIN3_MASK
0x3ff0_2e60	HT1_LO_WIN4_MASK	0x3ff0_2f60	HT1_HI_WIN4_MASK
0x3ff0_2e68	HT1_LO_WIN5_MASK	0x3ff0_2f68	HT1_HI_WIN5_MASK
0x3ff0_2e70	HT1_LO_WIN6_MASK	0x3ff0_2f70	HT1_HI_WIN6_MASK
0x3ff0_2e78	HT1_LO_WIN7_MASK	0x3ff0_2f78	HT1_HI_WIN7_MASK
0x3ff0_2e80	HT1_LO_WIN0_MMAP	0x3ff0_2f80	HT1_HI_WIN0_MMAP
0x3ff0_2e88	HT1_LO_WIN1_MMAP	0x3ff0_2f88	HT1_HI_WIN1_MMAP
0x3ff0_2e90	HT1_LO_WIN2_MMAP	0x3ff0_2f90	HT1_HI_WIN2_MMAP
0x3ff0_2e98	HT1_LO_WIN3_MMAP	0x3ff0_2f98	HT1_HI_WIN3_MMAP
0x3ff0_2ea0	HT1_LO_WIN4_MMAP	0x3ff0_2fa0	HT1_HI_WIN4_MMAP
0x3ff0_2ea8	HT1_LO_WIN5_MMAP	0x3ff0_2fa8	HT1_HI_WIN5_MMAP
0x3ff0_2eb0	HT1_LO_WIN6_MMAP	0x3ff0_2fb0	HT1_HI_WIN6_MMAP
0x3ff0_2eb8	HT1_LO_WIN7_MMAP	0x3ff0_2fb8	HT1_HI_WIN7_MMAP

二级 xbar 主要连接 2 个内存控制器和 IO-RING 作为从设备，由 4 个 Scache（4，代表 0x3ff0_4xxx，下同、5、6、7）和 IO-RING（9）作为主设备进行窗口映射，可以使用这些窗口配置寄存器（4、5、6、7、9）进行内存窗口配置，和地址转换。

每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 K 字节对齐，MASK 采用类似网络掩码高位为 1 的格式，MMAP 中包含转换后地址、路由选择及使能控制等位，如下表所示：

表 3-7MMAP 寄存器位域说明

[63:48]	[47：10]	[7：4]	[3：0]
保留	转换后地址	窗口使能	从设备号

其中，从设备号对应的设备如下表所示：

表 3-8 从设备号与所述模块的对应关系

从设备号	目的设备
0-3	Scache0-3
4-5	MC0-1
a	HT0_lo
b	HT0_hi
c	SE
d	MISC
e	HT1_lo
f	HT1_hi

窗口使能位的含义如下表所示：

表 3-9 MMAP 字段对应的该空间访问属性

[7]	[6]	[5]	[4]
窗口使能	允许对 DDR 进行交错访问，当从设备号为 0 时有效，按照“交错选择位 interleave_bit（CSR0x0400）”的配置对命中窗口地址的请求进行路由。要求交错使能位大于 10	允许块读	允许取指

需要注意的是，窗口配置不能对 Cache 一致性的请求进行地址转换，否则在 SCache 处的地址会与处理器一级 Cache 处的地址不一致，导致 Cache 一致性的维护错误。

窗口命中公式:　　(IN_ADDR & MASK) == BASE

新地址换算公式:　OUT_ADDR = (IN_ADDR & ~MASK) | {MMAP[63:10],10’h0}

根据缺省的寄存器配置，芯片启动后，CPU 的 0x00000000 - 0x0fffffff 的地址区间（256M）映射到 DDR 的 0x00000000 - 0x0fffffff 的地址区间，0x10000000-0x17ffffff 映射到桥片的 PCI_MEM 空间,0x18000000-0x19ffffff 映射到桥片的 PCI_IO 空间,0x1a000000-0x1affffff 映射到桥片的 PCI 配置空间（Type0），0x1b000000-0x1bffffff 映射到桥片的 PCI 配置空间（Type1），0x40000000-0x7fffffff 映射到桥片的 PCI_MEM 空间。软件可以通过修改相应的配置寄存器实现新的地址空间路由和转换。

此外，当出现由于 CPU 猜测执行引起对非法地址的读访问时，8 个地址窗口都不命中，将返回随机数据，以防止 CPU 死等。

4 芯片配置寄存器

龙芯 3A4000 中的芯片配置寄存器提供了对芯片的各种功能进行读写配置的机制。下面详述各个配置寄存器。

本章各个芯片配置寄存器的基地址为 0x1fe00000。

4.1 版本寄存器（0x0000）

基地址为 0x1fe00000，偏移地址 0x0000。

表 4-1 版本寄存器

位域	字段名	访问	复位值	描述
7:0	Version	R	8'h10	配置寄存器版本号

4.2 芯片特性寄存器（0x0008）

该寄存器标识了一些软件相关的处理器特性，供软件在使能特定功能前查看。寄存器的基地址为 0x1fe00000，偏移地址 0x0008。

表 4-2 芯片特性寄存器

位域	字段名	访问	复位值	描述
0	Centigrade	R	1'b1	为 1 时，表示 CSR[0x428]有效
1	Node counter	R	1'b1	为 1 时，表示 CSR[0x408]有效
2	MSI	R	1'b1	为 1 时，表示 MSI 可用
3	EXT_IOI	R	1'b1	为 1 时，表示 EXT_IOI 可用
4	IPI_percore	R	1'b1	为 1 时，表示通过 CSR 私有地址进行 IPI 发送
5	Freq_percore	R	1'b1	为 1 时，表示通过 CSR 私有地址调整频率
6	Freq_scale	R	1'b0	为 1 时，表示动态分频功能可用
7	DVFS_v1	R	1'b0	为 1 时，表示动态调频 v1 可用
8	Tsensor	R	1'b0	为 1 时，表示温度传感器可用

4.3 厂商名称（0x0010）

该寄存器用于标识厂商名称。基地址为 0x1fe00000，偏移地址 0x0010。

表 4-3 厂商名称寄存器

位域	字段名	访问	复位值	描述
63:0	Vendor	R	0x6e6f7367_6e6f6f4c	字符串“Loongson”

4.4 芯片名称（0x0020）

该寄存器用于标识芯片名称。基地址为 0x1fe00000，偏移地址 0x0020。

表 4-4 芯片名称寄存器

位域	字段名	访问	复位值	描述
63:0	ID	R	0x00003030_30344133	字符串“3A4000”

4.5 功能设置寄存器（0x0180）

基地址为 0x1fe00000，偏移地址 0x0180。

表 4-5 功能设置寄存器

位域	字段名	访问	复位值	描述
0		RW	1’b0	
1		RW	1’b0	
3:2		RW	2’b0	保留
4	MC0_disable_confspace	RW	1’b0	是否禁用 MC0 DDR 配置空间
5	MC0_defult_confspace	RW	1’b1	将所有内存访问路由至配置空间
6	MCA0 clock en	RW	1’b1	MCA0 时钟使能
7	MC0_resetn	RW	1’b1	MC0 软件复位（低有效）
8	MC0_clken	RW	1’b1	是否使能 MC0
9	MC1_disable_confspace	RW	1’b0	是否禁用 MC1 DDR 配置空间
10	MC1_defult_confspace	RW	1’b1	将所有内存访问路由至配置空间
11	MCA1 clock en	RW	1’b1	MCA1 时钟使能
12	MC1_resetn	RW	1’b1	MC1 软件复位（低有效）
13	MC1_clken	RW	1’b1	是否使能 MC1
26:24	HT0_freq_scale_ctrl	RW	3’b011	HT 控制器 0 分频
27	HT0_clken	RW	1’b1	是否使能 HT0
30:28	HT1_freq_scale_ctrl	RW	3’b011	HT 控制器 1 分频
31	HT1_clken	RW	1’b1	是否使能 HT1
42:40	Node_freq_ ctrl	RW	3’b111	结点分频
43	-	RW	1’b1	
63:56	Cpu_version	R	2’h3B	CPU 版本

4.6 引脚驱动设置寄存器（0x0188）

基地址为 0x1fe00000，偏移地址 0x0188。

表 4-6 引脚驱动设置寄存器

位域	字段名	访问	复位值	描述
15:0				（空）
17:16	HT sideband	RW	2’b0	HT 控制信号驱动设置
19:18	SE UART0	RW	2’b0	SE UART0 信号驱动设置
21:20	SE UART1	RW	2’b0	SE UART1 信号驱动设置
23:22	SE SPI	RW	2’b0	SE SPI 信号驱动设置
25:24	SE QSPI	RW	2’b0	SE QSPI 信号驱动设置
27:26	SE I2C	RW	2’b0	SE I2C 信号驱动设置
29:28	SE SCI	RW	2’b0	SE SCI 信号驱动设置
31:30	SE QSPI	RW	2’b0	SE QSPI 信号驱动设置
45:32				
47:46	SE RNG0	RW	2’b0	SE RNG0 信号驱动设置
49:48	SE RNG1	RW	2’b0	SE RNG1 信号驱动设置
51:50	SE GPIO	RW	2’b0	SE GPIO 信号驱动设置
53:52	UART	RW	2’b0	UART 信号驱动设置
55:54	SPI	RW	2’b0	SPI 信号驱动设置
57:56	GPIO	RW	2’b0	GPIO 信号驱动设置
59:58	I2C	RW	2’b0	I2C 信号驱动设置
61:60	SE CFG	RW	2’b0	SE CFG 信号驱动设置

4.7 功能采样寄存器（0x0190）

基地址为 0x1fe00000，偏移地址 0x0190。

表 4-7 功能采样寄存器

位域	字段名	访问	复位值	描述
31:0	Compcode_core	R		保留
37:32	Chip_config	R		主板配置控制
47:38	Sys_clkseli	R		板上倍频设置
55:48	Bad_ip_core	R		core7-core0 是否坏
57:56	Bad_ip_ddr	R		2 个 DDR 控制器是否坏
61:60	Bad_ip_ht	R		2 个 HT 控制器是否坏

4.8 温度采样寄存器（0x0198）

基地址为 0x1fe00000，偏移地址 0x0198。

表 4-8 温度采样寄存器

位域	字段名	访问	复位值	描述
15:0		R		保留
19:16	Compcode_ok	R		保留
20	dotest	R		Dotest 引脚状态
21	iccc_en	R		Iccc_en 引脚状态
23:22		R		保留
24	Thsens0_overflow	R		温度传感器 0 上溢
25	Thsens1_overflow	R		温度传感器 1 上溢
31:26				
47:32	Thsens0_out	R		温度传感器 0 摄氏温度 结点温度=Thens0_out *731/0x4000 - 273 温度范围 -40 度 – 125 度
63:48	Thsens1_out	R		温度传感器 1 摄氏温度 结点温度=Thens1_out *731/0x4000 - 273 温度范围 -40 度 – 125 度

4.9 偏压配置寄存器（0x01A0）

3A4000 内部集成了偏压产生模块，以下寄存器用于这些偏压模块的控制。基地址为 0x1fe00000，偏移地址为 0x1a0。

软件配置时，首先保持第 0 位为 0，将该寄存器其它位域设置为正确值，再将第 0 位设置为 1。

表 4-9 偏压设置寄存器

位域	字段名	访问	复位值	描述
0	BBGEN_enable	RW	0x0	偏压使能
1	BBMUX_first	RW	0x0	设置为先切换电压模式
3:2		RW	0x0	保留
7:4	BBGEN_feedback	RW	0x0	禁用 BBGEN 反馈信号
11:8	BBGEN_vbbp_val	RW	0x0	Vbbp 的设置值
15:12	BBGEN_vbbn_val	RW	0x0	Vbbn 的设置值
17:16	BBMUX_SEL_0	RW	0x0	BBMUX_SEL_0 设置值
19:18	BBMUX_SEL_1	RW	0x0	BBMUX_SEL_1 设置值

21:20	BBMUX_SEL_2	RW	0x0	BBMUX_SEL_2 设置值
23:22	BBMUX_SEL_3	RW	0x0	BBMUX_SEL_3 设置值
31:24		RW	0x0	保留
40:32	BBGEN_sm	RO	0x0	BBGEN 状态机当前状态
其它	-	RW		保留

4.10 频率配置寄存器（0x01B0）

以下几组软件倍频设置寄存器用于设置在 CLKSEL 配置为软件控制模式（参考 2.2 节的 CLKSEL 设置方法）下，芯片主时钟和内存控制器时钟的工作频率。其中，MEM CLOCK 配置对应内存控制器时钟频率，总线工作频率为该时钟的 2 倍，总线工作速率为该时钟的 4 倍；NODE CLOCK 对应处理器核、片上网络及高速共享缓存的时钟频率。

每个时钟配置一般有三个参数，DIV_REFC、DIV_LOOPC、DIV_OUT。最终的时钟频率为（参考时钟/DIV_REFC * DIV_LOOPC）/ DIV_OUT。

软件控制模式下，默认对应的时钟频率为外部参考时钟的频率（100MHz 或 25MHz），需要在处理器启动过程中对时钟进行软件设置。各个时钟设置的过程应该按照以下方式：

1) 设置寄存器中除了 SEL_PLL_*及 SOFT_SET_PLL 之外的其它寄存器，也即这两个寄存器在设置的过程中写为 0；
2) 其它寄存器值不变，将 SOFT_SET_PLL 设为 1；
3) 等待寄存器中的锁定信号 LOCKED_*为 1；
4) 将 SEL_PLL_*设为 1，此时对应的时钟频率将切换为软件设置的频率。

下面的寄存器为 Main CLOCK 的配置寄存器，Main Clock 用于产生 node clock、core clock 等的最高工作频率。其基地址为 0x1fe00000，偏移地址为 0x1b0：

表 4-10 结点时钟软件倍频设置寄存器

位域	字段名	访问	复位值	描述
0	SEL_PLL_NODE	RW	0x0	时钟输出选择 1：Node 时钟选择 PLL 输出 0：Node 时钟选择 SYS CLOCK
1		RW	0x0	保留
2	SOFT_SET_PLL	RW	0x0	允许软件设置 PLL
3	BYPASS_L1	RW	0x0	Bypass L1 PLL
15:4	-	RW	0x0	保留
16	LOCKED_L1	R	0x0	L1 PLL 是否锁定
18:17	-	R	0x0	保留
19	PD_L1	RW	0x0	关闭 L1 PLL
25:20		RW	0x0	保留
31:26	L1_DIV_REFC	RW	0x1	L1 PLL 输入参数

40:32	L1_DIV_LOOPC	RW	0x1	L1 PLL 输入参数
41				保留
47:42	L1_DIV_OUT	RW	0x1	L1 PLL 输入参数
				保留
其它	-	RW		保留

注：PLL ouput = （clk_ref /div_refc * div_loopc）/ div_out。

PLL 的 clk_ref/div_refc 的结果应该为 25 – 50MHz，而 VCO 频率（上述式中括号内部分）必须在范围 1.2GHz – 3.2GHz 之内。该要求对内存 PLL 同样适用。

下面的寄存器为 MEM CLOCK 的配置寄存器，MEM CLOCK 时钟频率应该配置为最终 DDR 总线时钟频率的 1/2。基地址为 0x1fe00000，偏移地址为 0x1c0：

表 4-11 内存时钟软件倍频设置寄存器

位域	字段名	访问	复位值	描述
0	SEL _MEM_PLL	RW	0x0	时钟输出选择 1：MEM 时钟选择 PLL 输出 0：MEM 时钟选择 SYS CLOCK
1	SOFT_SET_MEM_PLL	RW	0x0	允许软件设置 MEM PLL
2	BYPASS_MEM_PLL	RW	0x0	Bypass MEM_PLL
5:3				保留
6	LOCKED_MEM_PLL	R	0x0	MEM_PLL 是否锁定
7	PD_MEM_PLL	RW	0x0	关闭 MEM PLL
13:8	MEM_PLL_DIV_REFC	RW	0x1	MEM PLL 输入参数 当选用 NODE 时钟（NODE_CLOCK_SEL 为 1）时，作为分频输入
23:14	MEM_PLL_DIV_LOOPC	RW	0x41	MEM PLL 输入参数
29:24	MEM_PLL_DIV_OUT	RW	0x0	MEM PLL 输入参数
30	NODE_CLOCK_SEL	RW	0x0	0：使用 MEM_PLL 作为 MEM 时钟 1：使用 NODE_CLOCK 作为分频输入
其它		RW		保留

4.11 处理器核分频设置寄存器（0x01D0）

以下寄存器用于处理器核动态分频使用，使用该寄存器对处理器核进行调频设置，可以在 100ns 内完成变频操作，没有其它额外开销。基地址为 0x1fe00000，偏移地址 0x01d0。

表 4-12 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
2:0	core0_freqctrl	RW	0x7	核 0 分频控制值
3	core0_en	RW	0x1	核 0 时钟使能
6:4	core1_freqctrl	RW	0x7	核 1 分频控制值

7	core1_en	RW	0x1	核 1 时钟使能
10:8	core2_freqctrl	RW	0x7	核 2 分频控制值
11	core2_en	RW	0x1	核 2 时钟使能
14:12	core3_freqctrl	RW	0x7	核 3 分频控制值
15	core3_en	RW	0x1	核 3 时钟使能
			注:	软件分频后的时钟频率值等于原来的（分频控制值+1）/8

4.12 处理器核复位控制寄存器（0x01D8）

以下寄存器用于处理器核软件控制复位使用。需要复位时，先将对应核的 resetn 置 0，再将 resetn_pre 置 0，等待 500 微秒后，将 resetn_pre 置 1，再将 resetn 置 1 即可完成整个复位过程。该寄存器基地址为 0x1fe00000，偏移地址 0x01d8。

表 4-13 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
0	Core0_resetn_pre	RW	0x1	核 0 复位辅助控制
1	Core0_resetn	RW	0x1	核 0 复位
2	Core1_resetn_pre	RW	0x1	核 1 复位辅助控制
3	Core1_resetn	RW	0x1	核 1 复位
4	Core2_resetn_pre	RW	0x1	核 2 复位辅助控制
5	Core2_resetn	RW	0x1	核 2 复位
6	Core3_resetn_pre	RW	0x1	核 3 复位辅助控制
7	Core3_resetn	RW	0x1	核 3 复位

4.13 路由设置寄存器（0x0400）

以下寄存器用于控制芯片内的部分路由设置。基地址为 0x1fe00000，偏移地址 0x0400。

表 4-14 芯片路由设置寄存器

位域	字段名	访问	复位值	描述
3:0	scid_sel	RW	0x0	共享缓存散列位控制
6:4	Node_mask	RW	0x7	结点掩码，避免猜测到未使用结点的地址时无响应
7		RW	0x0	保留
8	xrouter_en	RW	0x0	HT1 片间路由使能控制
9	disable_0x3ff0	RW	0x0	禁止通过基地址 0x3ff0_0000 对配置寄存器空间的路由
12	mcc_en	RW	0x0	MCC 模式使能

19:16	ccsd_id	RW	0x0	
24	ccsd_en	RW	0x0	
31:30	mc_en	RW	0x3	使能两个 MC 的路由控制
37:32	interleave_bit	RW	0x0	内存散列位控制
39	interleave_en	RW	0x0	内存散列使能
43:40	ht_control	R		Ht 相关配置引脚
47:44	ht_reg_disable	RW	0x0	关闭 ht 空间，用于一致性模式下，避免 HT 空间地址路由到 HT

4.14 其它功能设置寄存器（0x0420）

以下寄存器用于控制芯片内部分功能使能。基地址为 0x1fe00000，偏移地址 0x0420。

表 4-15 其它功能设置寄存器

位域	字段名	访问	复位值	描述
0	disable_jtag	RW	0x0	完全禁用 JTAG 接口
1	disable_ejtag	RW	0x0	完全禁用 EJTAG 接口
2	disable_gs132	RW	0x0	完全禁用 GS132
3	disable_ejtag132	RW	0x0	完全禁用 GS132 EJTAG 接口
4	Disable_antifuse0	RW	0x0	禁用 fuse
5	Disable_antifuse1	RW	0x0	禁用 fuse
6	Disable_ID	RW	0x0	禁用 ID 修改
7				保留
8	resetn_gs132	RW	0x0	GS132 复位控制
9	sleeping_gs132	R	0x0	GS132 进入睡眠状态
10	soft_int_gs132	RW	0x0	GS132 核间中断寄存器
15:12	core_int_en_gs132	RW	0x0	GS132 对应每个核的 IO 中断使能
18:16	freqscale_gs132	RW	0x0	GS132 分频控制
19	clken_gs132	RW	0x0	GS132 时钟使能
21	stable_resetn	RW	0x0	稳定时钟复位控制
22	freqscale_percore	RW	0x0	使能每个核私有的调频寄存器
23	clken_percore	RW	0x0	使能每个核私有的时钟使能
27:24	confbus_timeout	RW	0x8	配置总线超时时间设置，实际时间为 2 的幂次方
29:28	HT_softresetn	RW	0x3	HT 控制器软件复位控制
35:32	freqscale_mode_core	RW	0x0	每个核的调频模式选择 0: (n+1)/8 1: 1/(n+1)
36	freqscale_mode_node	RW	0x0	结点的调频模式选择 0: (n+1)/8 1: 1/(n+1)

37	freqscale_mode_gs132	RW	0x0	GS132 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
39:38	freqscale_mode_HT	RW	0x0	每个 HT 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
43:41				保留
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能
57:56	thsensor_sel	RW	0x0	温度传感器选择
62:60	Auto_scale	R	0x0	自动调频当前值
63	Auto_scale_doing	R	0x0	自动调频正在生效标志

4.15 摄氏温度寄存器（0x0428）

以下寄存器用于观测芯片内部温度传感器数值。基地址为 0x1fe00000，偏移地址 0x0428。只有当 CSR[0x0008][0]有效时，该寄存器可用。

表 4-16 温度观测寄存器

位域	字段名	访问	复位值	描述
7:0	Centigrade temperature	RO	0x0	摄氏温度
63:8		RW	0x0	

4.16 SRAM 调节寄存器（0x0430）

以下寄存器用于调节处理器核内部 Sram 的工作频率。基地址为 0x1fe00000，偏移地址 0x0430。

表 4-17 处理器核 SRAM 调节寄存器

位域	字段名	访问	复位值	描述
31:0	sram_ctrl	RW	0x0	核内 Sram 配置寄存器
63:32		RW	0x0	

4.17 FUSE0 观测寄存器（0x0460）

以下寄存器用于观测部分软件可见的 Fuse0 数值。基地址为 0x1fe00000，偏移地址 0x0460。

表 4-18 FUSE 观测寄存器

位域	字段名	访问	复位值	描述
127:0	Fuse_0	RW	0x0	

4.18 FUSE1 观测寄存器（0x0470）

以下寄存器用于观测部分软件可见的 Fuse1 数值。基地址为 0x1fe00000，偏移地址 0x0470。

表 4-19 FUSE 观测寄存器

位域	字段名	访问	复位值	描述
127:0	Fuse_1	RW	0x0	

5 芯片时钟分频及使能控制

龙芯 3A4000 可以使用单一的外部参考时钟 SYS_CLOCK。各个时钟的产生都可以依赖于 SYS_CLOCK，下面章节对这些时钟分别介绍。

龙芯 3A4000 中为处理器核、片上网络及共享缓存、HT 控制器及 GS132 核分别设置了分频机制。相比原有 3A3000 的分频机制，在 3A4000 中实现的版本增加了新的分频模式，可以支持 1/n 的分频值。

本章各个芯片配置寄存器的基地址为 0x1fe00000。

5.1 芯片模块时钟介绍

芯片参考时钟 SYS_CLOCK 通常使用 100MHz 晶振输入，也可以选用 25MHz 晶振输入。不同晶振频率需要通过 CLKSEL[4]进行选择。

HT PHY 的参考时钟除了使用 SYS CLOCK,还可以使用每个 PHY 的 200MHz 差分参考输入。使用 CLKSEL[8]进行选择。当选用 SYS CLOCK 作为参考时钟，且使用 25MHz 晶振输入时，HT PHY 无法工作在 3.2GHz 的频率下。

龙芯 3A4000 芯片中所使用的时钟及其控制方式如下表所示。

表 5-1 处理器内部时钟说明

时钟	时钟来源	倍频方式	分频控制	使能控制	时钟描述
Boot Clock	SYS_CLOCK	*1	不支持	不支持	SPI、UART、I2C 控制器时钟
Main Clock	SYS PLL	PLL 配置	不支持	不支持	SYS PLL 输出。 Node Clock、Core Clock、HTcore Clock、GS132 Clock 时钟源 Mem Clock、Stable Clock 可选时钟源
Node Clock	Main Clock	*1	支持	不支持	片上网络、共享缓存、结点时钟、HT 控制器时钟源
Core0 Clock	Main Clock	*1	支持	支持	Core0 时钟
Core1 Clock	Main Clock	*1	支持	支持	Core1 时钟
Core2 Clock	Main Clock	*1	支持	支持	Core2 时钟
Core3 Clock	Main Clock	*1	支持	支持	Core3 时钟
HTcore0 Clock	Node Clock	*1	支持	支持	HT0 控制器时钟，软件需要保证分频后低于 1GHz
HTcore1 Clock	Node Clock	*1	支持	支持	HT1 控制器时钟，软件需要保证分频后低于 1GHz
GS132 Clock	Main Clock	*1	支持	支持	GS132 时钟，软件需要保证分频后低于 1GHz
Stable Clock	Main Clock	*1	支持	支持	处理器核恒定计数器时钟
Mem Clock	MEM PLL	PLL 配置	不支持	支持	内存控制器时钟

	Main Clock	/2、/4、/8	不支持	支持	内存控制器备选时钟

5.2 处理器核分频及使能控制

处理器核分频有多种模式，一为按地址访问模式，二是处理器配置指令访问模式，以下分别进行介绍。每个处理器核可以分别控制。

5.2.1 按地址访问

按地址访问模式与 3A3000 处理器兼容，使用处理器核软件分频设置寄存器，使用相同的地址进行设置。

使用该寄存器对处理器核进行调频设置，可以在 100ns 内完成变频操作，没有其它额外开销。基地址为 0x1fe00000，偏移地址 0x01d0。

表 5-2 处理器核软件分频设置寄存器

位域	字段名	访问	复位值	描述
2:0	core0_freqctrl	RW	0x7	核 0 分频控制值
3	core0_en	RW	0x1	核 0 时钟使能
6:4	core1_freqctrl	RW	0x7	核 1 分频控制值
7	core1_en	RW	0x1	核 1 时钟使能
10:8	core2_freqctrl	RW	0x7	核 2 分频控制值
11	core2_en	RW	0x1	核 2 时钟使能
14:12	core3_freqctrl	RW	0x7	核 3 分频控制值
15	core3_en	RW	0x1	核 3 时钟使能
			注:	软件分频后的时钟频率值等于原来的（分频控制值+1）/8

除了与 3A3000 处理器兼容的分频配置方式，3A4000 中还可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1fe00000，偏移地址 0x0420。

表 5-3 其它功能设置寄存器

位域	字段名	访问	复位值	描述
35:32	freqscale_mode_core	RW	0x0	每个核的调频模式选择 0: (n+1)/8 1: 1/(n+1)

5.2.2 配置寄存器指令访问

除了传统的按地址访问模式，3A4000 中还支持使用配置寄存器指令对私有的分频配置

寄存器进行访问。

需要注意的是，私有的分频配置寄存器控制与原有的处理器核软件分频设置寄存器控制是互斥的，两者只能选一种使用。选择的方法是通过“其它功能设置寄存器”上的对应位进行控制。该寄存器基地址为 0x1fe00000，偏移地址 0x0420。

表 5-4 其它功能设置寄存器

位域	字段名	访问	复位值	描述
22	freqscale_percore	RW	0x0	使能每个核私有的调频寄存器
23	clken_percore	RW	0x0	使能每个核私有的时钟使能

当 freqscale_percore 被设置为 1 时，使用私有的分频配置寄存器中的 freqscale 位对自己的时钟进行分频设置（包括了 freqscale_mode）；当 clken_percore 被设置为 1 时，使用私有的分频配置寄存器中的 clken 位对时钟使能进行控制。

该配置寄存器定义如下。偏移地址为 0x1050。

表 5-5 处理器核私有分频寄存器

位域	字段名	访问	复位值	描述
4	freqscale_mode	RW	0x0	当前处理器核的分频模式选择 0: (n+1)/8 1: 1/(n+1)
3	clken	RW	0x0	当前处理器核的时钟使能
2:0	freqscale	RW	0x0	当前处理器核的分频设置

5.3 结点时钟分频及使能控制

结点时钟为片上网络与共享缓存所使用的时钟，有两种不同的控制模式，一为软件设置模式，二是硬件自动分频设置。

结点时钟不支持完全关断功能，所以没有对应的 clken 控制位。

5.3.1 软件设置

软件设置方法与 3A3000 处理器兼容，使用功能设置寄存器中的结点分频位，使用相同的地址进行设置。

该寄存器基地址为 0x1fe00000，偏移地址 0x0180。

表 5-6 功能设置寄存器

位域	字段名	访问	复位值	描述
42:40	Node0_freq_ ctrl	RW	3’b111	结点 0 分频

与处理器核的分频控制一致，结点时钟也可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1fe00000，偏移地址 0x0420。

表 5-7 其它功能设置寄存器

位域	字段名	访问	复位值	描述
36	freqscale_mode_node	RW	0x0	结点的调频模式选择 0: (n+1)/8 1: 1/(n+1)

5.3.2 硬件自动设置

除了由软件进行主动的设置之外，结点时钟还支持由温度传感器触发的自动分频设置。自动分频设置是由软件预先针对不同的温度进行设置，当温度传感器的温度达到对应的预设值时，就会触发相应的自动分频设置。

为了在高温环境中保证芯片的运行，可以设置令高温自动降频，使得芯片在超过预设范围时主动进行时钟分频，达到降低芯片翻转率的效果。

对于高温降频功能，有 4 组控制寄存器对其行为进行设置。每组寄存器包含以下四个控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将触发分频操作；

EN：使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 3A4000 内部集成四个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。

FREQ：分频数。当触发分频操作时，这个分频数同样受到 freqscale_mode_node 的影响，当其为 0 时，将频率调整为当前时钟频率的(FREQ+1)/8 倍；为 1 时，将频率调整为当前时钟频率的 1/(FREQ+1)倍。

其基地址为 0x1fe00000 或 0x3ff00000。

表 5-8 高温降频控制寄存器说明

寄存器	地址	控制	说明
高温降频控制寄存器 Thsens_freq_scale	0x1480	RW	四组设置优先级由高到低 [7:0]：Scale_gate0：高温阈值 0，超过这个温度将降频 [8:8]：Scale_en0：高温降频使能 0 [11:10]：Scale_Sel0：选择高温降频 0 的温度传感器输入源 [14:12]：Scale_freq0：降频时的分频值 [23:16]：Scale_gate1：高温阈值 1，超过这个温度将降频 [24:24]：Scale_en1：高温降频使能 1 [27:26]：Scale_Sel1：选择高温降频 1 的温度传感器输入源 [30:28]：Scale_freq1：降频时的分频值 [39:32]：Scale_gate2：高温阈值 2，超过这个温度将降频 [40:40]：Scale_en2：高温降频使能 2 [43:42]：Scale_Sel2：选择高温降频 2 的温度传感器输入源 [46:44]：Scale_freq2：降频时的分频值 [55:48]：Scale_gate3：高温阈值 3，超过这个温度将降频 [56:56]：Scale_en3：高温降频使能 3 [59:58]：Scale_Sel3：选择高温降频 3 的温度传感器输入源 [62:60]：Scale_freq3：降频时的分频值
Thsens_freq_scale_up	0x1490	RW	温度传感器控制寄存器高位 [7:0] Scale_Hi_gate0 高 8 位 [15:8] Scale_Hi_gate1 高 8 位 [23:16] Scale_Hi_gate2 高 8 位 [31:24] Scale_Hi_gate3 高 8 位 [39:32] Scale_Lo_gate0 高 8 位 [47:40] Scale_Lo_gate1 高 8 位 [55:48] Scale_Lo_gate2 高 8 位 [63:56] Scale_Lo_gate3 高 8 位

5.4 HT 控制器分频及使能控制

HT 控制器的分频机制与其它的类似。两个 HT 控制器可以分别控制。使用功能设置寄存器中的对应位进行设置。其基地址为 0x1fe00000，偏移地址 0x0180。

表 5-9 功能设置寄存器

位域	字段名	访问	复位值	描述
26:24	HT0_freq_scale_ctrl	RW	3'b111	HT 控制器 0 分频

27	HT0_clken	RW	1'b1	是否使能 HT0
30:28	HT1_freq_scale_ctrl	RW	3'b111	HT 控制器 1 分频
31	HT1_clken	RW	1'b1	是否使能 HT1

与其它分频控制一致，HT 控制器时钟也可以通过寄存器的设置，将分频之后的时钟频率由原来的“（分频控制值+1）/8”调整为“1/（分频控制值+1）”。这个寄存器位于“其它功能设置寄存器”。基地址为 0x1fe00000，偏移地址 0x0420。

需要注意的是，因为 HT core clock 来源于 Node clock，因此也受到 Node clock 分频的影响。

表 5-10 其它功能设置寄存器

位域	字段名	访问	复位值	描述
39:38	freqscale_mode_HT	RW	0x0	HT 控制器的调频模式选择 0: (n+1)/8 1: 1/(n+1)

5.5 Stable Counter 分频及使能控制

Stable Counter 的分频机制与其它的类似。使用其它功能设置寄存器中的对应位进行设置。其基地址为 0x1fe00000，偏移地址 0x0420。

表 5-11 其它功能设置寄存器

位域	字段名	访问	复位值	描述
21	stable_reset	RW	0x0	稳定时钟复位控制 1：置为复位状态 0：解除软件复位
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能

需要注意的是，stable_reset 设置为 0 之后，只是解除了软件复位。此时，如果 GPIO_FUNC_en[13]为 1 时，stable counter 的复位还受到 GPIO[13]的控制（低有效）。

GPIO 输出使能寄存器基地址为 0x1fe00000，偏移地址 0x0500。

表 5-12 GPIO 输出使能寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_OEn	RW	32'hffffffff	GPIO 输出使能（低有效）
63:32	GPIO_FUNC_En	RW	32'hffff0000	GPIO 功能使能（低有效）

6 软件时钟系统

龙芯 3A4000 处理器中为系统软件使用的时钟定义了多个不同层次的使用方法。处理器核内部有传统的 counter/compare 寄存器，新增的 stable counter 寄存器，以及芯片级的 node counter 寄存器。

以下对 stable counter 和 node counter 进行介绍。

6.1 Stable Counter

龙芯 3A4000 中引入了一个新的恒定时钟源，称之为 stable counter。Stable counter 的时钟有别于处理器核自身的时钟，也有别于结点时钟，是一支独立的主时钟。

处理器核时钟与结点时钟都来源于主时钟，但都可以自由控制分频数（参见前一章的介绍），而 stable counter 的时钟同样来源于主时钟，也可以进行独立分频，不随其它时钟分频的变化而变化。

根据这个时钟源，实现了一个计时器和一个定时器。本章主要介绍 Stable couter 相关的寄存器。

6.1.1 Stable Timer 的配置地址

使用 Stable counter 时钟源，实现了一个单调增加的计时器 counter，和一个从设定值向下减的定时器 timer；每个处理器核有各自独立的 Stable counter 和 Stable timer。处理器访问计时器时，只能通过 rdhwr、DRDTIME 等特有指令来访问；处理器访问定时器时，可以通过地址来 load/store 访问，也可以通过 CSR 配置寄存器指令访问。

表 6-1 地址访问方式

名称	偏移地址	权限	描述
Core0_timer_config	0x1060	RW	处理器核 0 的定时器配置寄存器
Core0_timer_ticks	0x1070	R	处理器核 0 的定时器剩余值
Core1_timer_config	0x1160	RW	处理器核 1 的定时器配置寄存器
Core1_timer_ticks	0x1170	R	处理器核 1 的定时器剩余值
Core2_timer_config	0x1260	RW	处理器核 2 的定时器配置寄存器
Core2_timer_ticks	0x1270	R	处理器核 2 的定时器剩余值
Core3_timer_config	0x1360	RW	处理器核 3 的定时器配置寄存器
Core3_timer_ticks	0x1370	R	处理器核 3 的定时器剩余值

表 6-2 配置寄存器指令访问方式

名称	偏移地址	权限	描述
percore_timer_config	0x1060	RW	当前处理器核的定时器配置寄存器
percore_timer_ticks	0x1070	R	当前处理器核的定时器剩余值

表 6-3 寄存器含义

位域	字段名	访问	复位值	描述
timer_config				
63	1	RW	0x1	复位为 1，应写入为 1
62	Periodic	RW	0x0	循环计数使能。当该位为 1，则定时器减为 0 后，自动重置为 timer_config 中 InitVal 域的值。
61	Enable	RW	0x0	总使能。当该位为 1 时，定时器才生效。
47:0	InitVal	RW	0x0	进行倒计时的初始值
timer_ticks				
63:48	0	R	0x0	0 值
47:0	Ticks	R	0x0	倒计时的剩余值。当处于非循环计数时，计数完成后，该值将停留在 48'hffff_ffff_ffff。

6.1.2 Stable Counter 的时钟控制

Stable counter 使用主时钟，并且可以通过软件分频机制进行分频控制。

以下是 Stable counter 的时钟控制寄存器。该寄存器位于控制芯片其他功能设置寄存器。基地址为 0x1fe00000，偏移地址 0x0420。

表 6-4 其它功能设置寄存器

位域	字段名	访问	复位值	描述
21	stable_reset	RW	0x0	稳定时钟复位控制 1：置为复位状态 0：解除软件复位
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0: (n+1)/8 1: 1/(n+1)
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能

当 BIOS 对 Stable counter 时钟源进行配置后，需要更新每个处理器核中的 MCSR 部分用于控制 CPUCFG.0x4 和 CPUCFG.0x5 的值。参照第 8.1 节描述，CPUCFG.0x4 中应该填写以 Hz 为单位的晶振时钟频率；CPUCFG.0x5[31:16]应填写分频系数；CPUCFG.0x5[15:0]应填写倍频因子。后两者的填写，需要 BIOS 帮助进行计算，从而使得 CCFreq*CFM/CFD 的结果等于 Stable Counter 的实际频率。

6.1.3 Stable Counter 的校准

单芯片情况下，每个核的 Counter 差距在 2 个周期之内，无需特别的校准。在多芯片情况下，不同的芯片之间会有较大的差异，需要由一套专门的软硬件校准机制，将各个核的 counter 差异保持在 100ns 以下。

首先，为了保证每个芯片的主时钟在使用过程中不会产生偏差，使用同一个晶振驱动所有芯片的 SYS_CLK。

其次，为了保证每个芯片的 Stable counter 在同一时刻开始计时，硬件上需要使用两个 GPIO 管脚的复用功能。结点 0 使用 GPIO12 来输出复位信号，其它所有结点（包括结点 0）使用 GPIO13 来输入复位信号（需要配置为 Stable counter 功能）。在主板上需要使用缓冲器件来保证复位时序（主要是信号斜率），复位时序越好，不同芯片间的时钟差异越小。

软件在使用 Stable counter 之前必须对全局的 Stable counter 通过 GPIO12 来进行复位，复位之前需要保证各个芯片的时钟选择一致，各个芯片的复位已经解除。这个工作通常由 BIOS 来完成。系统的连接方案如下图所示。

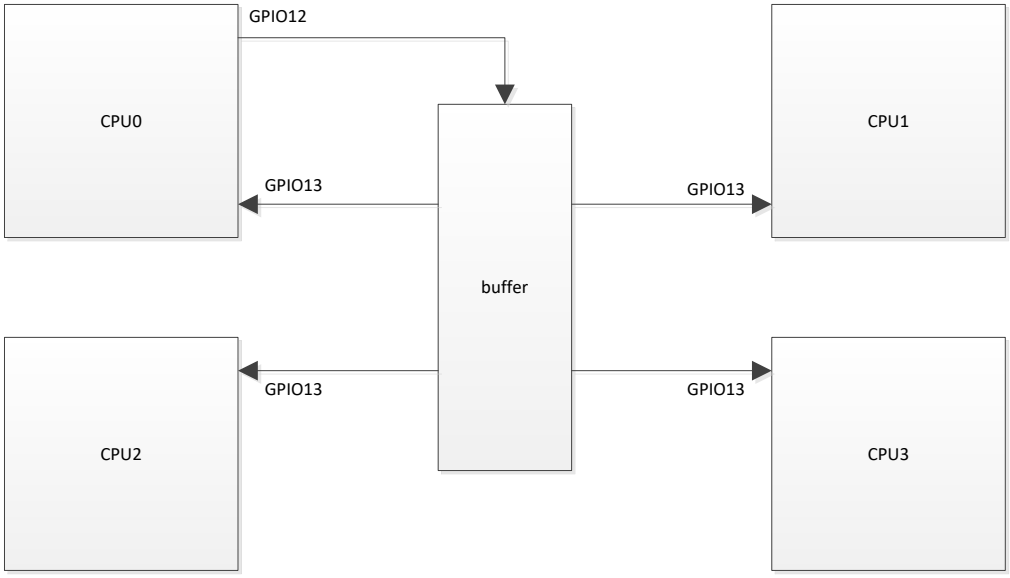


图 6-1 多片互连时的 Stable 复位控制

6.2 Node Counter

龙芯 3A4000 中的 Node counter 地址与 3A3000 及之前的芯片相同，但避免了原有的需要软件修正的问题，同时也可以采用配置寄存器指令进行访问。同样需要注意的是，与 3A3000 及之前的芯片相同，Node counter 的计数频率与 Node clock 完全相同，如果希望使用 Node counter 作为时钟计算依据，就要避免对 Node clock 进行变频。

6.2.1 按地址访问

按地址访问模式与 3A3000 处理器兼容，使用相同的地址进行设置。

配置寄存器的基地址为 0x1fe00000 或 0x3ff00000，如下表所示。

表 6-5 Node counter 寄存器

名称	偏移地址	权限	描述
Node counter	0x0408	R	64 位结点时钟计数

6.2.2 配置寄存器指令访问

Node counter 在使用配置寄存器指令进行访问时，与其它的配置寄存器略有不同。Node counter 的使用要求所有的处理器核访问同一个 counter，而不是各自的片上 counter（多片时），就要求每个核都访问同一个芯片的 node counter。因此，即使在多路系统中，每个芯片通过配置寄存器指令访问 CSR[0x408]，都是访问 NODE 0 上的 node counter。

具体的访问地址及寄存器定义请参考处理器核手册。

6.3 时钟系统小结

龙芯 3A4000 中新增的 Stable counter 在稳定性上比起 node counter 及 CP0 counter 更有优势，不会随着其它时钟（node clock 和 core clock）分频的变化而变化。

在易用性上来说 Stable counter 也更方便访问，使用 rdhwr 指令无论是用户态还是 Guest 态都能直接获得。Stable counter 是软件参考时钟系统的首选方案。

Node clock 更多是考虑传统兼容性的一个设计，是一个时钟系统的备份方案。

7 GPIO 控制

龙芯 3A4000 中提供最多 32 个 GPIO 供系统使用，绝大部分都与其它功能复用。通过寄存器设置，还可以将 GPIO 配置为中断输入功能，并可以设置其中断电平。

本章各个芯片配置寄存器的基地址为 0x1fe00000。

7.1 输出使能寄存器（0x0500）

基地址为 0x1fe00000，偏移地址 0x0500。

表 7-1 输出使能寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_OEn	RW	32'hffffffff	GPIO 输出使能（低有效）
63:32	GPIO_FUNC_En	RW	32'hffff0000	GPIO 功能使能（低有效）

7.2 输入输出寄存器（0x0508）

基地址为 0x1fe00000，偏移地址 0x0508。

表 7-2 输入输出寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_O	RW	32'h0	GPIO 输出设置
63:32	GPIO_I	RO	32'h0	GPIO 输入状态

7.3 中断控制寄存器（0x0510）

基地址为 0x1fe00000，偏移地址 0x0510。

表 7-3 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32'h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32'h0	GPIO 中断使能控制，高有效

7.4 GPIO 引脚功能复用表

3A4000 中 GPIO 引脚与其它功能进行了大量复用，以下列表为芯片功能引脚的引脚功能选择。

需要特别指出的是，GPIO00 – GPIO15 芯片复位时即为 GPIO 功能，默认为输入状态，不驱动 IO。

而 GPIO16 – GPIO31 是复用 HT 的各个控制引脚，复位时为 HT 功能，为了防止内部逻辑驱动对应的 IO，可以将对应的 HT0/1_Hi/Lo_Hostmode 引脚下拉。此时复位时虽然默认仍为 HT 功能，但却不会驱动 IO 引脚，不会对外部设备造成影响，只需要在软件使用 GPIO 功能前将功能设置为 GPIO 模式即可。

表 7-4 GPIO 功能复用表

GPIO 寄存器	引脚名称	复用功能	默认功能
0	GPIO00	SPI_CSn1	GPIO
1	GPIO01	SPI_CSn2	GPIO
2	GPIO02	UART1_RXD	GPIO
3	GPIO03	UART1_TXD	GPIO
4	GPIO04	UART1_RTS	GPIO
5	GPIO05	UART1_CTS	GPIO
6	GPIO06	UART1_DTR	GPIO
7	GPIO07	UART1_DSR	GPIO
8	GPIO08	UART1_DCD	GPIO
9	GPIO09	UART1_RI	GPIO
10	GPIO10	-	GPIO
11	GPIO11	-	GPIO
12	GPIO12	-	GPIO
13	GPIO13	SCNT_RSTn	GPIO
14	GPIO14	PROCHOTn	GPIO
15	GPIO15	THERMTRIPn	GPIO
16	HT0_LO_POWEROK	GPIO16	HT0_LO_POWEROK
17	HT0_LO_RSTn	GPIO17	HT0_LO_RSTn
18	HT0_LO_LDT_REQn	GPIO18	HT0_LO_LDT_REQn
19	HT0_LO_LDT_STOPn	GPIO19	HT0_LO_LDT_STOPn
20	HT0_HI_POWEROK	GPIO20	HT0_HI_POWEROK

21	HT0_HI_RSTn	GPIO21	HT0_HI_RSTn
22	HT0_HI_LDT_REQn	GPIO22	HT0_HI_LDT_REQn
23	HT0_HI_LDT_STOPn	GPIO23	HT0_HI_LDT_STOPn
24	HT1_LO_POWEROK	GPIO24	HT1_LO_POWEROK
25	HT1_LO_RSTn	GPIO25	HT1_LO_RSTn
26	HT1_LO_LDT_REQn	GPIO26	HT1_LO_LDT_REQn
27	HT1_LO_LDT_STOPn	GPIO27	HT1_LO_LDT_STOPn
28	HT1_HI_POWEROK	GPIO28	HT1_HI_POWEROK
29	HT1_HI_RSTn	GPIO29	HT1_HI_RSTn
30	HT1_HI_LDT_REQn	GPIO30	HT1_HI_LDT_REQn
31	HT1_HI_LDT_STOPn	GPIO31	HT1_HI_LDT_STOPn

7.5 GPIO 中断控制

3A4000 中 GPIO 引脚都可以作为中断输入使用。

GPIO00、GPIO08、GPIO16、GPIO24 共享中断控制器的 0 号中断线。

GPIO01、GPIO09、GPIO17、GPIO25 共享中断控制器的 1 号中断线。

GPIO02、GPIO10、GPIO18、GPIO26 共享中断控制器的 2 号中断线。

GPIO03、GPIO11、GPIO19、GPIO27 共享中断控制器的 3 号中断线。

GPIO04、GPIO12、GPIO20、GPIO28 共享中断控制器的 4 号中断线。

GPIO05、GPIO13、GPIO21、GPIO29 共享中断控制器的 5 号中断线。

GPIO06、GPIO14、GPIO22、GPIO30 共享中断控制器的 6 号中断线。

GPIO07、GPIO15、GPIO23、GPIO31 共享中断控制器的 7 号中断线。

每个 GPIO 的中断使能由配置寄存器 GPIO_INT_en 控制，中断电平由 GPIO_INT_POL 控制，寄存器如下：

基地址为 0x1fe00000，偏移地址 0x0510。

表 7-5 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32'h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32'h0	GPIO 中断使能控制，高有效

当中断控制器上的每个中断线只使能其中一位 GPIO 时，可以使用边沿触发方式，固定在某个沿（POL 设为 0 时下降沿，为 1 时上升沿）触发中断并在中断控制器中记录。

8 GS464V 处理器核

GS464V 是四发射 64 位的高性能处理器核。该处理器核既可以作为单核面向高端嵌入式应用和桌面应用，也可以作为基本的处理器核构成片内多核系统面向服务器和高性能机应用。在龙芯 3A4000 中的多个 GS464V 核通过以及共享 Cache 模块通过 AXI 互连网络形成一个分布式共享片上末级 Cache 的多核结构。GS464V 的主要特点如下：

- MIPS64 兼容，支持龙芯扩展指令集；
- 四发射超标量结构，四个定点、两个向量、两个访存部件；
- 每个向量部件宽度为 256bit，每个部件最多支持 8 个双 32 位浮点乘加运算；
- 访存部件支持 256 位存储访问，虚地址为 64 位，物理地址为 48 位；
- 支持寄存器重命名、动态调度、转移预测等乱序执行技术；
- 64 项全相联外加 8 路组相连 2048 项，共计 2112 项 TLB，64 项指令 TLB，可变页大小；
- 一级指令 Cache 和数据 Cache 大小各为 64KB，4 路组相联；
- Victim Cache 作为私有二级 Cache，大小为 256KB，16 路组相连；
- 支持 Non-blocking 访问及 Load-Speculation 等访存优化技术；
- 支持 Cache 一致性协议，可用于片内多核处理器；
- 一级 Cache 实现奇偶校验，二级、片上末级 Cache 实现 ECC 校验；
- 支持标准的 EJTAG 调试标准，方便软硬件调试；

GS464V 的结构如下图所示。

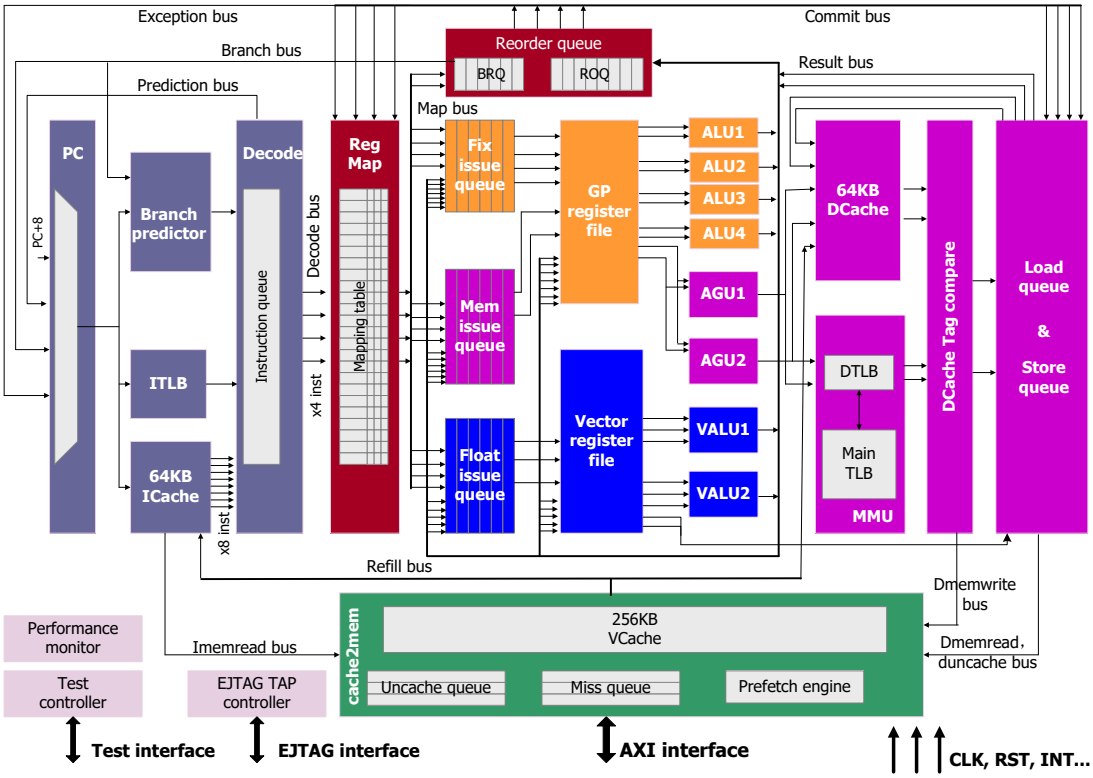


图 8-1 GS464V 结构图

8.1 3A4000 实现的指令集特性

龙芯 3A4000 具体实现的龙芯指令集功能特性，除了可以通过 MIPS 规范中已定义的方式进行识别，还可以通过龙芯指令集属性识别机制予以动态确认。

龙芯 3A4000 推荐软件使用龙芯自定义的 CPUCFG 指令进行龙芯指令集属性的识别(通过执行 RDCSR 读取相关 CSR 的方式也可以获得通用的信息，但是 RDCSR 只能在系统态下才能执行)。

CPUCFG 指令为用户态指令，其使用方式为 CPUCFG rd, rs，其中源操作数 rs 寄存器中存放待访问配置信息字的寄存器号，返回的配置字信息写入到 rd 寄存器中，每个配置信息字包含至多 32 位配置信息。例如，1 号配置字中包含了龙芯指令集中涉及 MIPS 兼容方面的相关信息，其中第 0 位表明是否支持硬件浮点协处理器，那么这个配置信息就表示为 CPUCFG.0x1.FP[bit0]，其中 0x1 表示配置信息字的字号是 1 号，FP 表示这个配置信息域的所起的助记名称叫 FP，bit0 表示 MSA1 这个域位于配置字的第 0 位。如果配置信息需要多位表达，那么其位置信息将会记为 bitAA:BB 的形式，表示从配置信息字的第 AA 位到第 BB 位的连续(AA-BB+1) 位。

下表给出 3A4000 实现的指令集功能配置信息列表。最后一列“可能取值”，表示从这个寄存器中有可能读出的值，但不意味着从 3A4000 处理器中读出的就是这个值。具体读出的值，请按照实际硬件执行该指令读出的结果为准，并按照实际读出的值，进行后续的软件判断，尽量不要按本表格最后一列的内容，直接断定某个 3A4000 芯片支持或不支持某功能。

表 8-1 3A4000 实现的指令集功能配置信息列表

寄存器号	位域	字段名	描述	可能取值
0x0	31:0	PRId	CP0.PRId	32’h14_8001
0x1	0	FP	等价于 CP0.Config1.FP[bit0]	1’b1
	3:1	FPRev	龙芯 FPU 浮点运算所遵循规范的版本号	3’h2
	4	MMI	为 1 表示实现了龙芯多媒体指令扩展	1’b1
	5			
	6			
	7			
	8			
	9	LSX1	为 1 表示支持龙芯 SIMD 扩展 I	1’b1
	10	LSX2	为 1 表示支持龙芯 SIMD 扩展 II	1’b1
	11	LASX	为 1 表示支持龙芯高级 SIMD 扩展	1’b1
	12			
	13			
	14			
	15	CNT64	为 1 表示 CP0.Count 为 64 位	1’b1
	16	LSLDR0	为 1 表示 load 到 R0 等效为预取的功能	1’b1
	17	LSPREF	为 1 表示 PREF 指令具有预取效果	1’b1
	18	LSPREFX	为 1 表示 PREFX 指令具有预取效果	1’b1
	19	LSSYNCI	为 1 表示 SYNCI 指令实现为串行化指令	1’b1
	20	LSUCA	为 1 表示支持用户态下执行部分 CACHE 指令	1’b1
	21	LLSYNC	为 1 表示需要在 LL 前加 SYNC 0 指令	1’b0
	22	TGTSYNC	为 1 表示 LL 与 SC 之间的分支需要在其跳转目标 处加 SYNC 0 指令	1’b0
	23	LLEXC	为 1 表示支持 LL 指令发起独占请求的功能	1’b1
	24	SCRAND	为 1 表示支持目录为 LL/SC 独占请求增加随机延 迟的功能	1’b1
	25	MUALP	为 1 表示支持非对齐访存功能	1’b1

	26	KMUALEn	为 1 表示在非用户态下非对齐访存功能已经开启	1’b0
	27	ITLBT	为 1 表示 ITLB 是软件透明的	1’b1
	28	LSUPERF	为 1 表示允许在用户态下用(D)MFC0 访问 Performance Counter	1’b1
	29	SFBP	为 1 表示支持 Store Fill Buffer 功能	1’b1
	30	CDMAP	为 1 表示支持 Cache DMA 功能	1’b1
0x2	0	LEXT1	为 1 表示实现了龙芯通用扩展 I	1’b1
	1	LEXT2	为 1 表示实现了龙芯通用扩展 II	1’b1
	2	LEXT3	为 1 表示实现了龙芯通用扩展 III	1’b1
	3	LSPW	为 1 表示实现了龙芯页表遍历指令扩展	1’b1
	4	LBT1	为 1 表示实现了龙芯二进制翻译加速扩展 I 版本	1’b1
	5	LBT2	为 1 表示实现了龙芯二进制翻译加速扩展 II 版本	1’b1
	6	LBT3	为 1 表示实现了龙芯二进制翻译加速扩展 III 版本	1’b1
	7	LBTMMU	为 1 表示实现了龙芯二进制翻译地址转换加速机制	1’b1
	8	LPMP	为 1 表示实现了龙芯性能计数器，此时 CP0.config1.PC[bit4]必然为 1	1’b1
	11:9	LPMRev	龙芯性能计数器实现版本号	3’h2
	13	LPIXU	为 1 表示支持启用用户态下龙芯位置无关扩展	1’b1
	14	LPIXNU	为 1 表示支持启用非用户态下龙芯位置无关扩展	1’b1
	15	LVZP	为 1 表示实现了龙芯虚拟化扩展	1’b1
	18:16	LVZRev	龙芯虚拟化规范的版本号	3’h2
	19	LGFTP	为 1 表示实现了全局恒定频率计时设备	1’b1
	22:20	LGFTPRev	全局恒定频率计时设备的版本号	3’h2
	23	LLFTP	为 1 表示实现了本地恒定频率计时设备	1’b1
	26:24	LLFTPRev	本地恒定频率计时设备的版本号	3’h2
	27	LCSRP	为 1 表示支持了龙芯控制状态寄存器	1’b1
	28	LDISBLIKELY	为 1 表示支持禁用 likely 分支指令的功能	1’b1
0x3	0	LCAMP	为 1 表示实现了硬件查找表功能	1’b1
	3:1	LCAMRev	硬件查找表功能的版本号	3’h2
	11:4	LCAMNUM	硬件查找表项数-1	8’h3f
	19:12	LCAMKW	硬件查找表 Key 域位宽-1	8’h2f
	27:20	LCAMVW	硬件查找表 Data 域位宽-1	8’h3f
0x4	31:0	CCFreq	处理器核晶振频率，单位 Hz	N/A
0x5	15:0	CFM	处理器核倍频因子	N/A

	31:16	CFD	处理器核分频系数	N/A
0x6	31:0	Safe	龙芯安全扩展参数	N/A
0x7	0	GCCAEQRP	为 1 表示支持 Guest CCA 仅有 Root 决定的功能	1'b1
	1	UCAWINP	为 1 表示支持非缓存加速属性由地址窗口配置功能	1'b1

8.2 3A4000 配置状态寄存器访问

3A4000 支持配置状态寄存器空间访问，CSR 使用一个新的独立的寻址空间进行访问，称为 CSR 空间，与现有的寄存器空间、内存空间和 EJTAG dseg 空间互不重叠。

CSR 通过自定义的 RDCSR 和 WRCSR 指令进行读、写访问。其中 RDCSR 的使用方式为 RDCSR rd, rs，其中源操作数 rs 寄存器中存放带访问的 CSR 的地址，CSR 读回的内容写入到 rd 寄存器中。WRCSR 的使用方式为 WRCSR rd, rs，其中源操作数 rs 寄存器中存放带访问的 CSR 的地址，源操作数 rd 寄存器中存放待写入 CSR 的值。RDCSR 和 WRCSR 只允许在核心态下运行。

使用 RDCSR/WRCSR 指令可以替代原有的地址映射配置寄存器的方式，即 0x1fe00000 和 0x3ff00000 空间，具体的访问方式参考相关章节说明。

此外，核内支持一组 CSR 寄存器，该组寄存器为每个处理器核所独有，寄存器说明如下。下列寄存器不能使用 0x3ff00000 和 0x1fe00000 空间访问。

表 8-2 核内配置状态寄存器列表

名称	地址	描述
GFTOffset	0xffffffffffffff8	Guest 模式下固定频率计时器偏移量
TimerID	0xffffffffffffff0	本地固定频率计时器的 ID 号
CSRffe8	0xffffffffffffffe8	调节参数，具体参见《龙芯 3A4000 指令系统手册》
ucacc_win0_lo	0xffffffffffffef8	非缓存加速窗口 0 的低位
ucacc_win1_lo	0xffffffffffffef0	非缓存加速窗口 1 的低位
ucacc_win2_lo	0xffffffffffffee8	非缓存加速窗口 2 的低位
ucacc_win3_lo	0xffffffffffffee0	非缓存加速窗口 3 的低位
ucacc_win0_hi	0xffffffffffffeb8	非缓存加速窗口 0 的高位
ucacc_win1_hi	0xffffffffffffeb0	非缓存加速窗口 1 的高位
ucacc_win2_hi	0xffffffffffffea8	非缓存加速窗口 2 的高位
ucacc_win3_hi	0xffffffffffffea0	非缓存加速窗口 3 的高位
MCSRWG	0xfffffffffff0000	MCSR 的写控制

9 共享 Cache（SCache）

SCache 模块是龙芯 3A4000 处理器内部所有处理器核所共享的三级 Cache。SCache 模块的主要特征包括：

- 采用 128 位 AXI 接口。
- 16 项 Cache 访问队列。
- 关键字优先。
- 通过目录支持 Cache 一致性协议。
- 可用于片上多核结构，也可直接和单处理器 IP 对接。
- 采用 16 路组相联结构。
- 支持 ECC 校验。
- 支持 DMA 一致性读写和预取读。
- 支持 16 种共享 Cache 散列方式。
- 支持按窗口锁共享 Cache。
- 保证读数据返回原子性。

共享 Cache 模块包括共享 Cache 管理模块 scachemanage 及共享 Cache 访问模块 scacheaccess。Scachemanage 模块负责处理器来自处理器和 DMA 的访问请求，而共享 Cache 的 TAG、目录和数据等信息存放在 scacheaccess 模块中。为降低功耗，共享 Cache 的 TAG、目录和数据可以分开访问，共享 Cache 状态位、w 位与 TAG 一起存储，TAG 存放在 TAG RAM 中，目录存放在 DIR RAM 中，数据存放在 DATA RAM 中。失效请求访问共享 Cache，同时读出所有路的 TAG、目录，并根据 TAG 来选出目录，并根据命中情况读取数据。替换请求、重填请求和写回请求只操作一路的 TAG、目录和数据。

为提高一些特定计算任务的性能，共享 Cache 增加了锁机制。落在被锁区域中的共享 Cache 块会被锁住，因而不会被替换出共享 Cache。通过芯片配置寄存器空间可以对共享 Cache 模块内部的四组锁窗口寄存器进行动态配置，锁 Cache 时连续区域大小需要小于 7.5MB。

表 9-1 共享 Cache 锁窗口寄存器配置

名称	地址	位域	描述
Slock0_valid	0x3ff00200	[63:63]	0 号锁窗口有效位
Slock0_addr	0x3ff00200	[47:0]	0 号锁窗口锁地址
Slock0_mask	0x3ff00240	[47:0]	0 号锁窗口掩码

Slock1_valid	0x3ff00208	[63:63]	1 号锁窗口有效位
Slock1_addr	0x3ff00208	[47:0]	1 号锁窗口锁地址
Slock1_mask	0x3ff00248	[47:0]	1 号锁窗口掩码
Slock2_valid	0x3ff00210	[63:63]	2 号锁窗口有效位
Slock2_addr	0x3ff00210	[47:0]	2 号锁窗口锁地址
Slock2_mask	0x3ff00250	[47:0]	2 号锁窗口掩码
Slock3_valid	0x3ff00218	[63:63]	3 号锁窗口有效位
Slock3_addr	0x3ff00218	[47:0]	3 号锁窗口锁地址
Slock3_mask	0x3ff00258	[47:0]	3 号锁窗口掩码

举例来说，当一个地址 addr 使得 slock0_valid && ((addr & slock0_mask) == (slock0_addr & slock0_mask))为 1 时，这个地址就被锁窗口 0 锁住了。

4 个 scache 使用同一个配置寄存器，基地址为 0x1fe00000，偏移地址 0x0280。

表 9-2 共享 Cache 配置寄存器（SC_CONFIG）

位域	字段名	访问	复位值	描述
0	LRU en	RW	1'b1	Scache LRU 替换算法使能
16	Prefetch En	RW	1'b1	Scache 预取功能使能
22:20	Prefetch config	RW	3'h1	当 scache 预取越过配置大小的地址范围时，停止预取 0 – 4KB 1 – 16KB 2 – 64KB 3 – 1MB 7 – 不受限 （注：SCID_SEL==0 时有效）
26:24	Prefetch lookahead	RW	3'h2	scache 预取步长 0 – 保留 1 – 0x100 2 – 0x200 3 – 0x300 4 – 0x400 5 – 0x500 6 – 0x600 7 – 0x700 （注：SCID_SEL==0 时有效）
30:28	Sc stall dirq cycle	RW	3'h2	SC 指令堵住 dirq 的时钟周期数 0 – 1 cycle（nonstall） 1 – 16-31 cycle random 2 – 32-63 cycle random 3- 64-127 cycle random

				4 – 128-255 cycle random 其他 – 无效值
31	MCC storefill en	RW	1’b0	MCC storefill 功能使能

10 处理器核间中断与通信

龙芯 3A4000 为每个处理器核都实现了 8 个核间中断寄存器（IPI）以支持多核 BIOS 启动和操作系统运行时在处理器核之间进行中断和通信。

龙芯 3A4000 中支持两种不同的访问方式，一种是与 3A3000 等处理器兼容的地址访问模式，另一种是为了支持处理器寄存器空间的直接私有访问。后面章节进行分别说明。

10.1 按地址访问模式

对于龙芯 3A4000，下列寄存器可以使用基地址 0x3ff0_0000 或者 0x1fe0_0000 进行访问。其中，基地址 0x3ff0_0000 可以通过路由设置寄存器中的 disable_0x3ff0 控制位进行关闭。具体寄存器说明和地址见表 10-1 到表 10-5。

表 10-1 处理器核间中断相关的寄存器及其功能描述

名称	读写权限	描述
IPI_Status	R	32 位状态寄存器，任何一位有被置 1 且对应位使能情况下，处理器核 INT4 中断线被置位。
IPI_Enable	RW	32 位使能寄存器，控制对应中断位是否有效
IPI_Set	W	32 位置位寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被置 1
IPI_Clear	W	32 位清除寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被清 0
MailBox0	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox01	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox02	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox03	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。

在龙芯 3A4000 与处理器核间中断相关的寄存器及其功能描述如下：

表 10-2 0 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core0_IPI_Status	0x1000	R	0 号处理器核的 IPI_Status 寄存器
Core0_IPI_Enalbe	0x1004	RW	0 号处理器核的 IPI_Enalbe 寄存器
Core0_IPI_Set	0x1008	W	0 号处理器核的 IPI_Set 寄存器
Core0_IPI_Clear	0x100c	W	0 号处理器核的 IPI_Clear 寄存器

Core0_MailBox0	0x1020	RW	0 号处理器核的 IPI_MailBox0 寄存器
Core0_ MailBox1	0x1028	RW	0 号处理器核的 IPI_MailBox1 寄存器
Core0_ MailBox2	0x1030	RW	0 号处理器核的 IPI_MailBox2 寄存器
Core0_ MailBox3	0x1038	RW	0 号处理器核的 IPI_MailBox3 寄存器

表 10-3 1 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core1_IPI_Status	0x1100	R	1 号处理器核的 IPI_Status 寄存器
Core1_IPI_Enalbe	0x1104	RW	1 号处理器核的 IPI_Enalbe 寄存器
Core1_IPI_Set	0x1108	W	1 号处理器核的 IPI_Set 寄存器
Core1_IPI_Clear	0x110c	W	1 号处理器核的 IPI_Clear 寄存器
Core1_MailBox0	0x1120	R	1 号处理器核的 IPI_MailBox0 寄存器
Core1_ MailBox1	0x1128	RW	1 号处理器核的 IPI_MailBox1 寄存器
Core1_ MailBox2	0x1130	W	1 号处理器核的 IPI_MailBox2 寄存器
Core1_ MailBox3	0x1138	W	1 号处理器核的 IPI_MailBox3 寄存器

表 10-4 2 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core2_IPI_Status	0x1200	R	2 号处理器核的 IPI_Status 寄存器
Core2_IPI_Enalbe	0x1204	RW	2 号处理器核的 IPI_Enalbe 寄存器
Core2_IPI_Set	0x1208	W	2 号处理器核的 IPI_Set 寄存器
Core2_IPI_Clear	0x120c	W	2 号处理器核的 IPI_Clear 寄存器
Core2_MailBox0	0x1220	R	2 号处理器核的 IPI_MailBox0 寄存器
Core2_ MailBox1	0x1228	RW	2 号处理器核的 IPI_MailBox1 寄存器
Core2_ MailBox2	0x1230	W	2 号处理器核的 IPI_MailBox2 寄存器
Core2_ MailBox3	0x1238	W	2 号处理器核的 IPI_MailBox3 寄存器

表 10-5 3 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core3_IPI_Status	0x1300	R	3 号处理器核的 IPI_Status 寄存器
Core3_IPI_Enalbe	0x1304	RW	3 号处理器核的 IPI_Enalbe 寄存器
Core3_IPI_Set	0x1308	W	3 号处理器核的 IPI_Set 寄存器
Core3_IPI_Clear	0x130c	W	3 号处理器核的 IPI_Clear 寄存器
Core3_MailBox0	0x1320	R	3 号处理器核的 IPI_MailBox0 寄存器
Core3_ MailBox1	0x1328	RW	3 号处理器核的 IPI_MailBox1 寄存器
Core3_ MailBox2	0x1330	W	3 号处理器核的 IPI_MailBox2 寄存器
Core3_ MailBox3	0x1338	W	3 号处理器核的 IPI_MailBox3 寄存器

上面列出的是单个龙芯 3A4000 芯片所组成的单结点多处理器系统的的核间中断相关寄存器列表。在采用多片龙芯 3A4000 互连构成多结点 CC-NUMA 系统时，每个芯片内的结点对应一个系统全局结点编号，结点内处理器核的 IPI 寄存器地址按上表与其所在结点的基地址成固定偏移关系。例如，0 号结点 0 号处理器核的 IPI_Status 地址为 0x3ff01000，而 1 号结点的 0 号处理器地址则为 0x10003ff01000，依次类推。

10.2 配置寄存器指令访问

在龙芯 3A4000 中，新增了处理器核直接的寄存器访问指令，可以通过私有空间对配置寄存器进行访问。为了更方便地使用核间中断寄存器，在这个模式下，对核间中断寄存器定义进行一些调整。

表 10-6 当前处理器核核间中断与通信寄存器列表

名称	偏移地址	权限	描述
perCore_IPI_Status	0x1000	R	当前处理器核的 IPI_Status 寄存器
perCore_IPI_Enalbe	0x1004	RW	当前处理器核的 IPI_Enalbe 寄存器
perCore_IPI_Set	0x1008	W	当前处理器核的 IPI_Set 寄存器
perCore_IPI_Clear	0x100c	W	当前处理器核的 IPI_Clear 寄存器
perCore_MailBox0	0x1020	RW	当前处理器核的 IPI_MailBox0 寄存器
perCore_ MailBox1	0x1028	RW	当前处理器核的 IPI_MailBox1 寄存器
perCore_ MailBox2	0x1030	RW	当前处理器核的 IPI_MailBox2 寄存器
perCore_ MailBox3	0x1038	RW	当前处理器核的 IPI_MailBox3 寄存器

为了向其它核发送核间中断请求及 MailBox 通信，通过以下寄存器进行访问。

表 10-7 处理器核核间通信寄存器

名称	偏移地址	权限	描述
IPI_Send	0x1040	WO	32 位中断分发寄存器 [31] 等待完成标志，置 1 时会等待中断生效 [30:26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 中断向量号，对应 IPI_Status 中的向量
Mail_Send	0x1048	WO	64 位 MailBox 缓存寄存器 [63:32] MailBox 数据 [31] 等待完成标志，置 1 时会等待写入生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入

			[26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:2] MailBox 号 0 - MailBox0 低 32 位 1 - MailBox0 高 32 位 2 - MailBox1 低 32 位 3 - MailBox1 高 32 位 4 - MailBox2 低 32 位 5 - MailBox2 高 32 位 6 - MailBox3 低 32 位 7 - MailBox4 高 32 位 [1:0] 保留
FREQ_Send	0x1058	WO	32 位频率使能寄存器 [31] 等待完成标志，置 1 时会等待设置生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 写入对应的处理器核私有频率配置寄存器。CSR[0x1050]

需要注意的是，由于 Mail_Send 寄存器一次只可以发送 32 位的数据，当发送 64 位数据时必须拆分为两次发送。因此，目标核在等待 Mail_Box 内容时，需要通过其它的软件手段来确保传输的完整性。例如，发送完 Mail_Box 数据之后，通过核间中断来表示已经发送完成。

11 I/O 中断

龙芯 3A4000 芯片支持两种不同的中断方式。第一种为传统中断方式，与 3A3000 等处理器兼容；第二种为新增的扩展 IO 中断方式，用于支持 HT 控制器的中断跨片与动态分发功能。以下分别对两种中断方式进行介绍。

11.1 传统 I/O 中断

龙芯 3A4000 芯片的传统中断支持 32 个中断源，以统一方式进行管理，如下图所示。任意一个 IO 中断源可以被配置为是否使能、触发的方式、以及被路由的目标处理器核中断脚。传统中断不支持中断的跨片分发，只能中断同一个处理器片内的处理器核。

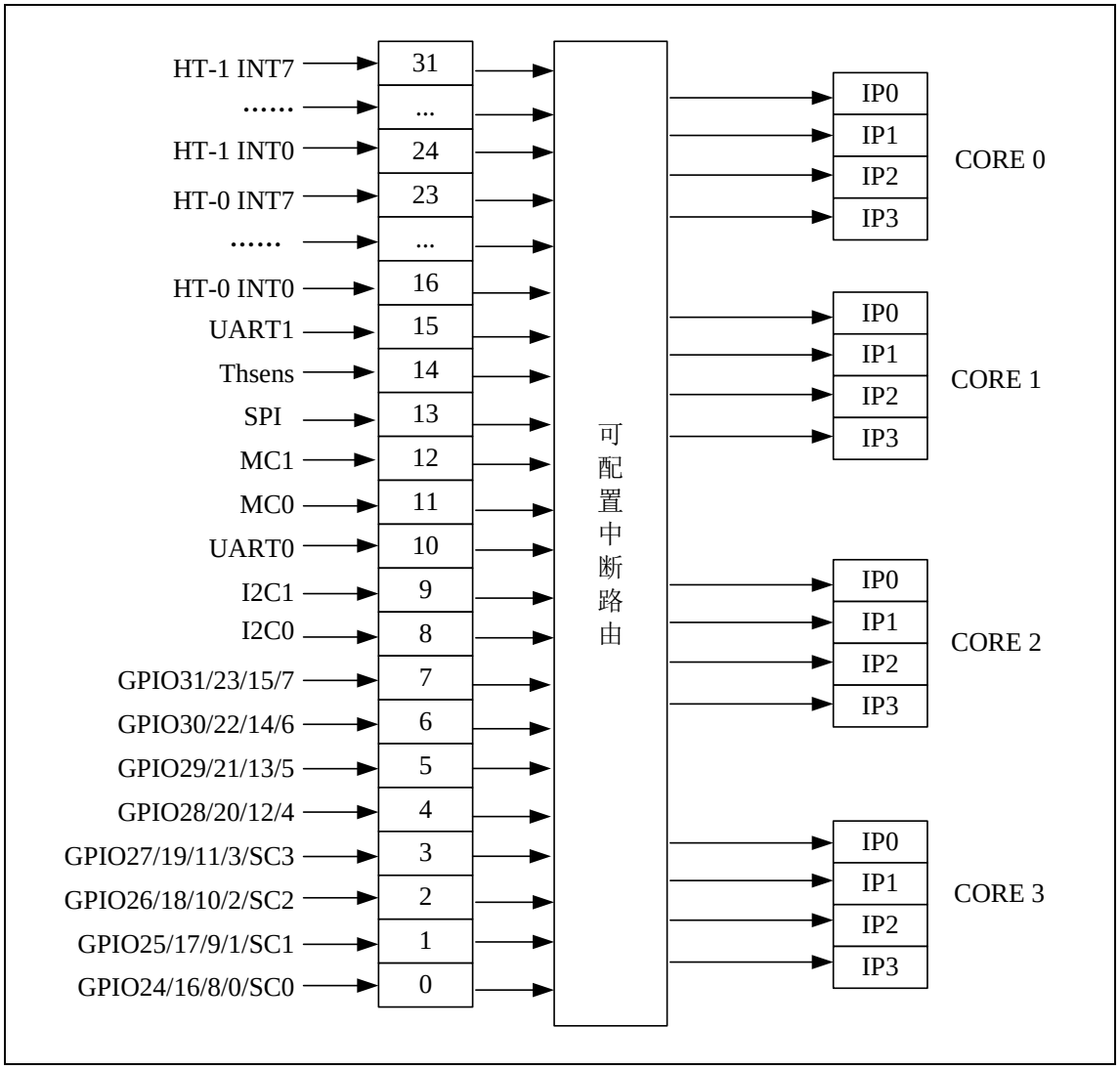


图 11-1 龙芯 3A4000 处理器中断路由示意图

中断相关配置寄存器都是以位的形式对相应的中断线进行控制，中断控制位连接及属性配置见下表。

中断使能（Enable）的配置有三个寄存器：Intenset、Intenclr 和 Inten。Intenset 设置中断使能，Intenset 寄存器写 1 的位对应的中断被使能。Intenclr 清除中断使能，Intenclr 寄存器写 1 的位对应的中断被清除。Inten 寄存器读取当前各中断使能的情况。

边沿触发的中断信号由 Intedge 配置寄存器来选择，写 1 表示边沿触发，写 0 表示电平触发。中断处理程序可以通过 Intenclr 的相应位来清除中断记录，清除中断的同时也会清除中断使能。

表 11-1 中断控制寄存器

位域	访问属性/缺省值				
	Intedge	Inten	Intenset	Intenclr	中断源
0	RW / 0	R / 0	RW / 0	RW / 0	GPIO24/16/8/0/SC0
1	RW / 0	R / 0	RW / 0	RW / 0	GPIO25/17/9/1/SC1
2	RW / 0	R / 0	RW / 0	RW / 0	GPIO26/18/10/2/SC2
3	RW / 0	R / 0	RW / 0	RW / 0	GPIO27/19/11/3/SC3
4	RW / 0	R / 0	RW / 0	RW / 0	GPIO28/20/12/4
5	RW / 0	R / 0	RW / 0	RW / 0	GPIO29/21/13/5
6	RW / 0	R / 0	RW / 0	RW / 0	GPIO30/22/14/6
7	RW / 0	R / 0	RW / 0	RW / 0	GPIO31/23/15/7
8	RW / 0	R / 0	RW / 0	RW / 0	I2C0
9	RW / 0	R / 0	RW / 0	RW / 0	I2C1
10	RW / 0	R / 0	RW / 0	RW / 0	UART0
11	RW / 0	R / 0	RW / 0	RW / 0	MC0
12	RW / 0	R / 0	RW / 0	RW / 0	MC1
13	RW / 0	R / 0	RW / 0	RW / 0	SPI
14	RW / 0	R / 0	RW / 0	RW / 0	Thsens
15	RW / 0	R / 0	RW / 0	RW / 0	UART1
23 : 16	RW / 0	R / 0	RW / 0	RW / 0	HT0[7:0]
31 : 24	RW / 0	R / 0	RW / 0	RW / 0	HT1[7:0]

与核间中断类似，IO 中断的基地址同样可以使用 0x1fe00000 或 0x3ff00000 进行访问，也可以通过处理器核的专用寄存器配置指令进行访问。

11.1.1 按地址访问

这种访问方式与 3A3000 等处理器的访问方式兼容，基地址可以使用 0x1fe00000 或

0x3ff00000。0x3ff00000 的基地址可以通过路由配置寄存器中的 disable_0x3ff0 控制位进行禁用。

表 11-2 IO 控制寄存器地址

名称	偏移地址	描述
Intisr	0x1420	32 位中断状态寄存器
Inten	0x1424	32 位中断使能状态寄存器
Intenset	0x1428	32 位设置使能寄存器
Intenclr	0x142c	32 位清除使能寄存器
Intedge	0x1434	32 位触发方式寄存器
CORE0_INTISR	0x1440	路由给 CORE0 的 32 位中断状态
CORE1_INTISR	0x1448	路由给 CORE1 的 32 位中断状态
CORE2_INTISR	0x1450	路由给 CORE2 的 32 位中断状态
CORE3_INTISR	0x1458	路由给 CORE3 的 32 位中断状态

龙芯 3A4000 中集成了 4 个处理器核，上述的 32 位中断源可以通过软件配置选择期望中断的目标处理器核。进一步，中断源可以选择路由到处理器核中断 INT0 到 INT3 中的任意一个，即对应 CP0_Status 的 IP2 到 IP5。32 个 I/O 中断源中每一个都对应一个 8 位的路由控制器，其格式和地址如表 11-3 和表 11-4 所示。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 3 号处理器的 INT2 上。

表 11-3 中断路由寄存器的说明

位域	说　明
3:0	路由的处理器核向量号
7:4	路由的处理器核中断引脚向量号

表 11-4 中断路由寄存器地址

名称	偏移地址	描述	名称	偏移地址	描述
Entry0	0x1400	GPIO24/16/8/0	Entry16	0x1410	HT0-int0
Entry1	0x1401	GPIO25/17/9/1	Entry17	0x1411	HT0-int1
Entry2	0x1402	GPIO26/18/10/2	Entry18	0x1412	HT0-int2
Entry3	0x1403	GPIO27/19/11/3	Entry19	0x1413	HT0-int3
Entry4	0x1404	GPIO28/20/12/4	Entry20	0x1414	HT0-int4
Entry5	0x1405	GPIO29/21/13/5	Entry21	0x1415	HT0-int5
Entry6	0x1406	GPIO30/22/14/6	Entry22	0x1416	HT0-int6
Entry7	0x1407	GPIO31/23/15/7	Entry23	0x1417	HT0-int7

Entry8	0x1408	I2C0	Entry24	0x1418	HT1-int0
Entry9	0x1409	I2C1	Entry25	0x1419	HT1-int1
Entry10	0x140a	UART0	Entry26	0x141a	HT1-int2
Entry11	0x140b	MC0	Entry27	0x141b	HT1-int3
Entry12	0x140c	MC1	Entry28	0x141c	HT1-int4
Entry13	0x140d	SPI	Entry29	0x141d	HT1-int5
Entry14	0x140e	Thsens	Entry30	0x141e	HT1-int6
Entry15	0x140f	UART1	Entry31	0x141f	HT1-int7

11.1.2 配置寄存器指令访问

在龙芯 3A4000 中，同样可以通过配置寄存器指令的访问方法，通过私有空间对配置寄存器进行访问。指令所使用的偏移地址与通过地址访问的方式相同。此外，为了方便用户的使用，对每个核不同的当前中断状态设置了专用的私有中断状态寄存器，如下表所示。

表 11-5 处理器核私有中断状态寄存器

名称	偏移地址	描述
perCore_INTISR	0x1010	路由给当前处理器核的 32 位中断状态

11.2 扩展 I/O 中断

除了兼容原有的传统 IO 中断方式，3A4000 开始支持扩展 I/O 中断，用于将 HT 总线上的 256 位中断直接分发给各个处理器核，而不再通过 HT 的中断线进行转发，提升 IO 中断使用的灵活性。

内核在使用扩展 IO 中断前，需要使能“其他功能设置寄存器”中的对应位。该寄存器基地址为 0x1fe00000，偏移地址 0x0420。

表 11-6 其它功能设置寄存器

位域	字段名	访问	复位值	描述
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能

在扩展 IO 中断模式下，HT 中断可以直接进行跨片的转发以及轮转分发等操作。当前版本，最多可以支持 256 个扩展中断向量。

11.2.1 按地址访问

以下是相关的扩展 IO 中断寄存器。与其它的配置寄存器一样，基地址可以使用 0x1fe00000 或 0x3ff00000，也可以通过处理器核的专用寄存器配置指令进行访问。

表 11-7 扩展 IO 中断使能寄存器

名称	偏移地址	描述
EXT_IOIen[63:0]	0x1600	扩展 IO 中断[63:0]的中断使能配置
EXT_IOIen[127:64]	0x1608	扩展 IO 中断[127:64]的中断使能配置
EXT_IOIen[191:128]	0x1610	扩展 IO 中断[191:128]的中断使能配置
EXT_IOIen[255:192]	0x1618	扩展 IO 中断[255:192]的中断使能配置

表 11-8 扩展 IO 中断自动轮转使能寄存器

名称	偏移地址	描述
EXT_IOIbounce[63:0]	0x1680	扩展 IO 中断[63:0]的自动轮转使能配置
EXT_IOIbounce[127:64]	0x1688	扩展 IO 中断[127:64]的自动轮转使能配置
EXT_IOIbounce[191:128]	0x1690	扩展 IO 中断[191:128]的自动轮转使能配置
EXT_IOIbounce[255:192]	0x1698	扩展 IO 中断[255:192]的自动轮转使能配置

表 11-9 扩展 IO 中断状态寄存器

名称	偏移地址	描述
EXT_IOIsr[63:0]	0x1700	扩展 IO 中断[63:0]的中断状态
EXT_IOIsr[127:64]	0x1708	扩展 IO 中断[127:64]的中断状态
EXT_IOIsr[191:128]	0x1710	扩展 IO 中断[191:128]的中断状态
EXT_IOIsr[255:192]	0x1718	扩展 IO 中断[255:192]的中断状态

表 11-10 各处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
CORE0_EXT_IOIsr[63:0]	0x1800	路由至处理器核 0 的扩展 IO 中断[63:0]的中断状态
CORE0_EXT_IOIsr[127:64]	0x1808	路由至处理器核 0 的扩展 IO 中断[127:64]的中断状态
CORE0_EXT_IOIsr[191:128]	0x1810	路由至处理器核 0 的扩展 IO 中断[191:128]的中断状态
CORE0_EXT_IOIsr[255:192]	0x1818	路由至处理器核 0 的扩展 IO 中断[255:192]的中断状态
CORE1_EXT_IOIsr[63:0]	0x1900	路由至处理器核 1 的扩展 IO 中断[63:0]的中断状态
CORE1_EXT_IOIsr[127:64]	0x1908	路由至处理器核 1 的扩展 IO 中断[127:64]的中断状态
CORE1_EXT_IOIsr[191:128]	0x1910	路由至处理器核 1 的扩展 IO 中断[191:128]的中断状态

CORE1_EXT_IOIsr[255:192]	0x1918	路由至处理器核 1 的扩展 IO 中断[255:192]的中断状态
CORE2_EXT_IOIsr[63:0]	0x1A00	路由至处理器核 2 的扩展 IO 中断[63:0]的中断状态
CORE2_EXT_IOIsr[127:64]	0x1A08	路由至处理器核 2 的扩展 IO 中断[127:64]的中断状态
CORE2_EXT_IOIsr[191:128]	0x1A10	路由至处理器核 2 的扩展 IO 中断[191:128]的中断状态
CORE2_EXT_IOIsr[255:192]	0x1A18	路由至处理器核 2 的扩展 IO 中断[255:192]的中断状态
CORE3_EXT_IOIsr[63:0]	0x1B00	路由至处理器核 3 的扩展 IO 中断[63:0]的中断状态
CORE3_EXT_IOIsr[127:64]	0x1B08	路由至处理器核 3 的扩展 IO 中断[127:64]的中断状态
CORE3_EXT_IOIsr[191:128]	0x1B10	路由至处理器核 3 的扩展 IO 中断[191:128]的中断状态
CORE3_EXT_IOIsr[255:192]	0x1B18	路由至处理器核 3 的扩展 IO 中断[255:192]的中断状态

与传统 IO 中断类似，扩展 IO 中断的 256 位中断源也可以通过软件配置选择期望中断的目标处理器核。

但中断源并不可以单独选择路由到处理器核中断 INT0 到 INT3 中的任意一个，而是以组为单位进行 INT 中断的路由，以中断对应 CP0_Status 的 IP2 到 IP5。下面是按组进行配置的中断引脚路由寄存器。

表 11-11 中断引脚路由寄存器的说明

位域	说　明
3:0	路由的处理器核中断引脚向量号
7:4	保留

表 11-12 中断路由寄存器地址

名称	偏移地址	描述
EXT_IOImap0	0x14C0	EXT_IOI[31:0]的引脚路由方式
EXT_IOImap1	0x14C1	EXT_IOI[63:32]的引脚路由方式
EXT_IOImap2	0x14C2	EXT_IOI[95:64]的引脚路由方式
EXT_IOImap3	0x14C3	EXT_IOI[127:96]的引脚路由方式
EXT_IOImap4	0x14C4	EXT_IOI[159:128]的引脚路由方式
EXT_IOImap5	0x14C5	EXT_IOI[191:160]的引脚路由方式
EXT_IOImap6	0x14C6	EXT_IOI[223:192]的引脚路由方式
EXT_IOImap7	0x14C7	EXT_IOI[255:224]的引脚路由方式

每个中断源都另外对应一个 8 位的路由控制器，其格式和地址如下表 11-13 和表 11-14 所示。其中[7:4]用于在表 11-15 中选择真正的结点路由向量。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 EXT_IOI_node_type4 所指结点的 3 号处理器核。

表 11-13 中断目标处理器核路由寄存器的说明

位域	说　明
3:0	路由的处理器核向量号
7:4	路由的结点映射方式选择（如表 11-15 的配置方法）

需要注意的是，当使用轮转分发模式时（对应的 EXT_IOIbounce 为 1），在结点号与处理器核号的全映射模式上轮转。EXT_IOIbounce 的设置应该在相关的路由映射配置之后。

例如，当表 11-13 中的设置为 0x27，而表 11-15 中的 EXT_IOI_node_type2 的设置为 0x0013 时，使能轮转分发模式下，该中断将依次在节点 0 核 0、节点 0 核 1、节点 0 核 2、节点 1 核 0、节点 1 核 1、节点 1 核 2、节点 4 核 0、节点 4 核 1、节点 4 核 2 上轮转。

当使用固定分发模式时（对应的 EXT_IOIbounce 为 0），结点号的 bitmap 上只允许有一位为 1，或为全 0，对应本地触发。

表 11-14 中断目标处理器核路由寄存器地址

名称	偏移地址	描述
EXT_IOImap_Core0	0x1C00	EXT_IOI[0]的处理器核路由方式
EXT_IOImap_Core1	0x1C01	EXT_IOI[1]的处理器核路由方式
EXT_IOImap_Core2	0x1C02	EXT_IOI[2]的处理器核路由方式
……		
EXT_IOImap_Core254	0x1CFE	EXT_IOI[254]的处理器核路由方式
EXT_IOImap_Core255	0x1CFF	EXT_IOI[255]的处理器核路由方式

表 11-15 中断目标结点映射方式配置

名称	偏移地址	描述
EXT_IOI_node_type0	0x14A0	16 个结点的映射向量类型 0（软件配置）
EXT_IOI_node_type1	0x14A2	16 个结点的映射向量类型 1（软件配置）
EXT_IOI_node_type2	0x14A4	16 个结点的映射向量类型 2（软件配置）
……		
EXT_IOI_node_type15	0x14BE	16 个结点的映射向量类型 15(软件配置）

11.2.2 配置寄存器指令访问

使用处理器核的配置寄存器指令进行访问时，最大的不同在于对处理器核的中断状态寄存器的访问成为私有的访问，每个核都只需要向同一个地址发出查询请求就可以得到当前核的中断状态。

表 11-16 当前处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
perCore_EXT_IOIsr[63:0]	0x1800	路由至当前处理器核的扩展 IO 中断[63:0]的中断状态
perCore_EXT_IOIsr[127:64]	0x1808	路由至当前处理器核的扩展 IO 中断[127:64]的中断状态
perCore_EXT_IOIsr[191:128]	0x1810	路由至当前处理器核的扩展 IO 中断[191:128]的中断状态
perCore_EXT_IOIsr[255:192]	0x1818	路由至当前处理器核的扩展 IO 中断[255:192]的中断状态

11.2.3 扩展 IO 中断触发寄存器

为了支持扩展 IO 中断的动态分发，在配置寄存器中增加了一个扩展 IO 中断触发寄存器，用于将对应的 IO 中断置位。平时可以利用这个寄存器对中断进行调试或测试。

这个寄存器的说明如下：

表 11-17 扩展 IO 中断触发寄存器

名称	偏移地址	权限	描述
EXT_IOI_send	0x1140	WO	扩展 IO 中断设置寄存器 [7:0]为期望设置的中断向量

11.2.4 扩展 IO 中断与传统 HT 中断处理的区别

传统的 HT 中断处理方式下，HT 中断由 HT 控制器进行内部处理，直接映射到 HT 配置寄存器上的 256 个中断向量，再由 256 个中断向量分组产生 4 个或 8 个中断，再路由至各个不同的处理器核。由于采用的是传统的中断线连接，不能直接产生跨片中断，由此所有的 HT IO 中断都只能直接由单个芯片进行处理。另一方面，芯片内硬件分发的中断只是以最终的 4 个或 8 个中断为单位，不能按位处理，由此导致硬件中断分发不好用的问题。

扩展 IO 中断方式，HT 中断由 HT 控制器直接发给芯片的中断控制器进行处理，中断控制器能直接得到 256 位中断，而不是之前的 4 个或 8 个中断，这 256 位中断每一位都可以独立路由，独立分发，而且可以实现跨片的分发及轮转。

使用扩展 IO 中断之后，软件处理上与使用传统的 HT 中断稍有不同。

传统的HT中断处理时，内核直接到HT控制器的中断向量（一般为0x90000efdfb000080）上进行查找，然后按位进行处理，此时无论路由模式如何配置，都是直接读到 HT 控制器上的所有中断。

使用扩展 IO 中断之后，内核直接到扩展 IO 状态寄存器（配置空间 0x1800）上读取中断状态进行处理，每个核只会读到中断自己的中断状态并进行处理，不同核之间不会产生干扰。

12 温度传感器

12.1 实时温度采样

龙芯 3A4000 内部集成两个温度传感器，可以通过 0x1FE00198 开始的采样寄存器进行观测，同时，可以使用灵活的高低温中断报警或者自动调频功能进行控制。温度传感器在采样寄存器的对应位如下（基地址为 0x1FE00000，偏移地址为 0x0198）：

表 12-1 温度采样寄存器说明

位域	字段名	访问	复位值	描述
24	Thsens0_overflow	R		温度传感器 0 上溢
25	Thsens1_overflow	R		温度传感器 1 上溢
47:32	Thsens0_out	R		温度传感器 0 摄氏温度 结点温度=Thens0_out *731/0x4000 - 273 温度范围 -40 度 – 125 度
65:48	Thsens1_out	R		温度传感器 1 摄氏温度 结点温度=Thens1_out -*731/0x4000 - 273 温度范围 -40 度 – 125 度

通过对控制寄存器的设置，可以实现超过预设温度中断、低于预设温度中断及高温自动降频功能。

此外，还可以使用新增的摄氏温度寄存器直接读取当前的摄氏温度。这个寄存器同样可以使用 0x1FE00000 或者 0x3FF00000 为基地址的读操作进行访问，也可以使用配置寄存器指令进行直接访问，偏移地址为 0x0428。该寄存器描述如下：

表 12-2 扩展 IO 中断触发寄存器

名称	偏移地址	权限	描述
Thsens_Temperature	0x0428	R	温度传感器摄氏温度

12.2 高低温中断触发

对于高低温中断报警功能，分别有 4 组控制寄存器对其阈值进行设置。每组寄存器包含以下三个控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将会产生中断。需要注意的是，Gate 值的设置应该是与 0x198 寄存器相对应的 16 位数值，而不是摄氏温度；

EN：中断使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 3A4000 内部集成两个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。可以使用 0 或者 1。

高温中断控制寄存器中包含 4 组用于控制高温中断触发的设置位；低温中断控制寄存器中包含 4 组用于控制低温中断触发的设置位。另外还有一组寄存器用于显示中断状态，分别对应于高温中断和低温中断，对该寄存器进行任意写操作将清除中断状态。

这几个寄存器的具体描述如下，其基地址为 0x1fe00000 或 0x3ff00000：

表 12-3 高低温中断寄存器说明

寄存器	地址	控制	说明
高温中断控制寄存器 Thsens_int_ctrl_Hi	0x1460	RW	[7:0]：Hi_gate0：高温阈值 0，超过这个温度将产生中断 [8:8]：Hi_en0：高温中断使能 0 [11:10]：Hi_Sel0：选择高温中断 0 的温度传感器输入源 [23:16]：Hi_gate1：高温阈值 1，超过这个温度将产生中断 [24:24]：Hi_en1：高温中断使能 1 [27:26]：Hi_Sel1：选择高温中断 1 的温度传感器输入源 [39:32]：Hi_gate2：高温阈值 2，超过这个温度将产生中断 [40:40]：Hi_en2：高温中断使能 2 [43:42]：Hi_Sel2：选择高温中断 2 的温度传感器输入源 [55:48]：Hi_gate3：高温阈值 3，超过这个温度将产生中断 [56:56]：Hi_en3：高温中断使能 3 [59:58]：Hi_Sel3：选择高温中断 3 的温度传感器输入源
低温中断控制寄存器 Thsens_int_ctrl_Lo	0x1468	RW	[7:0]：Lo_gate0：低温阈值 0，低于这个温度将产生中断 [8:8]：Lo_en0：低温中断使能 0 [11:10]：Lo_Sel0：选择低温中断 0 的温度传感器输入源 [23:16]：Lo_gate1：低温阈值 1，低于这个温度将产生中断 [24:24]：Lo_en1：低温中断使能 1 [27:26]：Lo_Sel1：选择低温中断 1 的温度传感器输入源 [39:32]：Lo_gate2：低温阈值 2，低于这个温度将产生中断 [40:40]：Lo_en2：低温中断使能 2 [43:42]：Lo_Sel2：选择低温中断 2 的温度传感器输入源 [55:48]：Lo_gate3：低温阈值 3，低于这个温度将产生中断 [56:56]：Lo_en3：低温中断使能 3 [59:58]：Lo_Sel3：选择低温中断 3 的温度传感器输入源
中断状态寄存器 Thsens_int_status/clr	0x1470	RW	中断状态寄存器，写 1 清除中断 [0]：高温中断触发 [1]：低温中断触发
高低温中断控制高位 Thsens_int_up	0x1478	RW	[7:0] Hi_gate0 高 8 位 [15:8] Hi_gate1 高 8 位 [23:16] Hi_gate2 高 8 位 [31:24] Hi_gate3 高 8 位 [39:32] Lo_gate0 高 8 位 [47:40] Lo_gate1 高 8 位 [55:48] Lo_gate2 高 8 位 [63:56] Lo_gate3 高 8 位

12.3 高温自动降频设置

为了在高温环境中保证芯片的运行，可以设置令高温自动降频，使得芯片在超过预设范围时主动进行时钟分频，达到降低芯片翻转率的效果。

对于高温降频功能，有 4 组控制寄存器对其行为进行设置。每组寄存器包含以下四个

控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将触发分频操作；

EN：使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 3A4000 内部集成四个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入。

FREQ：分频数。当触发分频操作时，使用预设的 FREQ 对时钟进行分频，分频的模式受 freqscale_mode_node 的控制。

其基地址为 0x1fe00000 或 0x3ff00000。

表 12-4 高温降频控制寄存器说明

寄存器	地址	控制	说明
高温降频控制寄存器 Thsens_freq_scale	0x1480	RW	四组设置优先级由高到低 [7:0]：Scale_gate0：高温阈值 0，超过这个温度将降频 [8:8]：Scale_en0：高温降频使能 0 [11:10]：Scale_Sel0：选择高温降频 0 的温度传感器输入源 [14:12]：Scale_freq0：降频时的分频值 [23:16]：Scale_gate1：高温阈值 1，超过这个温度将降频 [24:24]：Scale_en1：高温降频使能 1 [27:26]：Scale_Sel1：选择高温降频 1 的温度传感器输入源 [30:28]：Scale_freq1：降频时的分频值 [39:32]：Scale_gate2：高温阈值 2，超过这个温度将降频 [40:40]：Scale_en2：高温降频使能 2 [43:42]：Scale_Sel2：选择高温降频 2 的温度传感器输入源 [46:44]：Scale_freq2：降频时的分频值 [55:48]：Scale_gate3：高温阈值 3，超过这个温度将降频 [56:56]：Scale_en3：高温降频使能 3 [59:58]：Scale_Sel3：选择高温降频 3 的温度传感器输入源 [62:60]：Scale_freq3：降频时的分频值
Thsens_freq_scale_up	0x1490	RW	温度传感器控制寄存器高位 [7:0] Scale_Hi_gate0 高 8 位 [15:8] Scale_Hi_gate1 高 8 位 [23:16] Scale_Hi_gate2 高 8 位 [31:24] Scale_Hi_gate3 高 8 位 [39:32] Scale_Lo_gate0 高 8 位 [47:40] Scale_Lo_gate1 高 8 位 [55:48] Scale_Lo_gate2 高 8 位 [63:56] Scale_Lo_gate3 高 8 位

12.4 温度状态检测与控制

引脚 PROCHOTn 和 THERMTRIPn 用于温度状态检测与控制，这两个信号分别与 GPIO14 和 GPIO15 复用。其中 PROCHOTn 既可作为输入也可作为输出，THERMTRIPn 仅有输出功能。

PROCHOTn 作为输入时，芯片受外部温度检测电路的控制，外部温度检测电路需要降低芯片温度时可以置 PROCHOTn 为 0，芯片接收到该低电平后会采取降频措施，降频时的分频值由通过寄存器 prochotn_freq_scale 设置。PROCHOTn 作为输出时，芯片可输出高温中断，通过 prochotn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

THERMTRIPn 作为输出，由芯片通过 thermtripn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

虽然 THERMTRIPn 和 PROCHOTn 都是对外的高温中断，但是 THERMTRIPn 的紧急程度较 PROCHOTn 更高。PROCHOTn 置位时，外部温度控制电路还可以采取一定的措施，比如提高风扇转速。而 THERMTRIPn 置位时，外部电源控制电路应该直接采取紧急断电措施。

具体的控制寄存器如下：

表 12-5 温度状态检测与控制寄存器说明

寄存器	地址	控制	说明
温度状态检测与控制寄存器 Thsens_hi_ctrl	0x1498	RW	[0:0]：prochotn_oe PROCHOTn 引脚输出使能控制，0 为输出，1 为输入 [5:4]：prochotn_o_sel PROCHOTn 高温中断输出选择 [10:8]：prochotn_freq_scale：PROCHOTn 输入有效时的分频值 [17:16]：thermtripn_o_sel THERMTRIPn 高温中断输出选择

12.5 温度传感器的控制

3A4000 内部集成了 4 个温度传感器，可通过寄存器配置调整温度/电压监测，监测点配置与监测频率等配置，还可直接观测到每一个温度传感器的输出内容用于调试。（基地址为 0x1FE00000，温度传感器配置寄存器的偏移地址为 0x01580+vtsensor_id<<4， 温度传感器数据寄存器的偏移地址为 0x01588+vtsensor_id<<4）

表 12-6 温度传感器配置寄存器说明

位域	字段名	访问	复位值	描述
0	Thsens_trigger	RW	0	使能温度传感器配置，如置位，可由 thsens_mode 和 thsens_cluster 选择监测模式和监测点；为 0 则默认温度监测模式，且监测点由

				temp_cluster 配置。
2	Thsens_mode	RW	0	0：温度模式；1：电压模式
3	Thsens_datarate	RW	0	监测频率： 0 – 10~20Hz 1 – 325~650Hz
6:4	Thsens_cluster	RW	0	传感器监测点配置：0 为本地监测点，1~7 为远程监测点
8	Temp_valid	RW	0	使能温度传感器输出，替换 CSR[0x198] 中 Thsens0_out 和 Thsens0_overflow 的值为该温度传感器的温度监测值。
11:9	Temp_cluster	RW	0	温度传感器输出监测点选择，Thsens_trigger 使能时无效

表 12-7 温度传感器数据寄存器说明

位域	字段名	访问	复位值	描述
3	Out_mode	R	0	传感器配置的监测模式 0：温度模式；1：电压模式
6:4	Out_cluster	R	0	传感器配置的监测点
7	Overflow	R	0	传感器监测值溢出
29:16	Data	R	0	传感器读出的监测值

读出值的计算方法：

结点温度=data*731/0x4000 - 273（温度范围 -40 度 ~ 125 度）

电压=data*1.226/0x1000

监测点的配置如下

表 12-8 温度传感器监测点说明

传感器	Cluster	监测点	传感器	Cluster	监测点
0	0	Reserved	2	0	Reserved
	1	Core0 监测点 0		1	Core2 监测点 0
	2	Core0 监测点 1		2	Core2 监测点 1
	3	Core0 监测点 2		3	Scache2
	4	Core0 监测点 3		4	Mc1-phy 监测点 0
	5	SCache0		5	Mc0-phy 监测点 0
	6	HT0		6	Mc0-ctrl
	7	Reserved		7	Reserved
1	0	Reserved	3	0	Reserved
	1	Core1 监测点 0		1	Core3 监测点 2
	2	Core1 监测点 1		2	Core3 监测点 3
	3	Core1 监测点 2		3	Scache3

	4	SCache1		4	Mc0-phy 监测点 1
	5	L1X		5	Mc1-phy 监测点 1
	6	HT1		6	Mc1-ctrl
	7	NOC-VERT		7	L2X

13 DDR3/4 SDRAM 控制器配置

龙芯 3A4000 处理器内部集成的内存控制器的设计遵守 DDR3/4 SDRAM 行业标准（JESD79-3 和 JESD79-4）。在龙芯 3A4000 处理器中，所实现的所有内存读/写操作都遵守 JESD79-3 及 JESD79-4 的规定。

13.1 DDR3/4 SDRAM 控制器功能概述

龙芯 3A4000 处理器支持 DDP 和 3DS 封装模式。其中 DDP 最大支持 8 个 CS（由 8 个 DDR3/DDR4 SDRAM 片选信号实现，即 4 个双面内存条），3DS 最大支持 4 个 CS（由 8 个 DDR4 SDRAM 片选信号实现，即 32 个逻辑 RANK）。一共含有 22 位的地址总线（即：18 位的行列地址总线、2 位逻辑 Bank 总线和 2 位逻辑 Bank Group 总线，其中行列地址总线与 RASn、CASn 和 Wen 复用）。

龙芯 3A4000 处理器在具体选择使用不同内存芯片类型时，可以调整 DDR3/4 控制器参数设置进行支持。其中，支持的最大片选（CS_n）为 8，逻辑 RANK（CHIP ID）数为 8，行地址（ROW）数为 18，列地址（COL）数为 12，逻辑体选择（BANK）数为 2（DDR4）或 3（DDR3），逻辑体组（BANK Group）数为 2（仅 DDR4）。其中 DDR3 和 DDR4 的管脚有复用关系，具体见下表。另外 CS_n 与 Chip ID 的复用关系可配，具体请参考 13.4 小节。

表 13-1 DDR3/4 地址控制信号复用

PAD 名称	DDR3	DDR4
DDR_ACTn	DDR_A15	DDR_ACTn
DDR_RASn	DDR_RASn	DDR_RASn/DDR_A16
DDR_CASn	DDR_CASn	DDR_CASn/DDR_A15
DDR_WEn	DDR_WEn	DDR_WEn/DDR_A14
DDR_BG[1]	DDR_A14	DDR_BG1
DDR_BG[0]	DDR_BA[2]	DDR_BG0

CPU 发送的内存请求物理地址可以根据控制器内部不同的配置进行多种不同的地址映射。

龙芯 3A4000 处理器所集成的内存控制电路只接受来自处理器或者外部设备的内存读/写请求，在所有的内存读/写操作中，内存控制电路处于从设备状态（Slave State）。

龙芯 3A4000 处理器中内存控制器具有如下特征：

- 接口上命令、读写数据全流水操作；
- 内存命令合并、排序提高整体带宽；
- 配置寄存器读写端口，可以修改内存设备的基本参数；
- 内建动态延迟补偿电路（DCC），用于数据的可靠发送和接收；

- ECC 功能可以对数据通路上的 1 位和 2 位错误进行检测，并能对 1 位错误进行自动纠错；
- 支持 DDR3/4 SDRAM，且参数配置支持 x4、x8、x16 颗粒；
- 控制器与 PHY 频率比 1/2 ；
- 支持数据传输速率范围为 800Mbps-3200Mbps。

13.2 DDR3/4 SDRAM 读操作协议

DDR3 SDRAM 读操作的协议如图 13-1 所示。在图中命令（Command，简称 CMD）由 RAS_n、CAS_n 和 WE_n 共 3 个信号组成。对于读操作，RAS_n=1，CAS_n=0，WE_n=1。

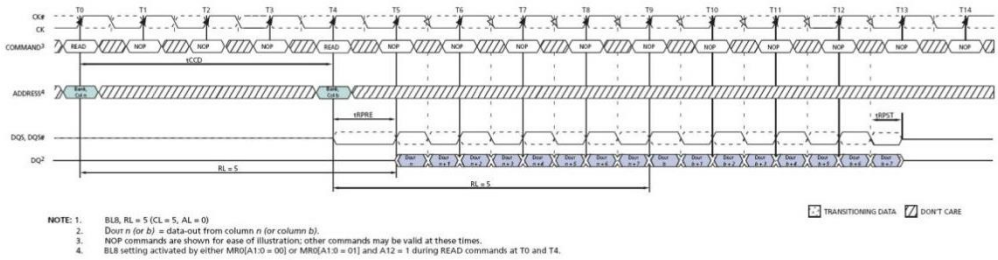


图 13-1 DDR3 SDRAM 读操作协议

上图中，Cas Latency（CL） = 5， Read Latency（RL）=5， Burst Length = 8。

DDR4 SDRAM 读操作协议类似。在图中命令 CMD 由 ACT_n、RAS_n、CAS_n 和 WE_n 共 4 个信号组成。对于读操作，ACT_n=1，RAS_n=1，CAS_n=0，WE_n=1。

13.3 DDR3/4 SDRAM 写操作协议

DDR3 SDRAM 写操作的协议如图 13-2 所示。在图中命令 CMD 由 RAS_n、CAS_n 和 WE_n 共 3 个信号组成。对于写操作，RAS_n=1，CAS_n=0，WE_n=0。另外，与读操作不同，写操作可以通过 DQM 来标识写操作的数据掩码，即需要写入的字节数，DQM 与图中 DQS 信号同步。

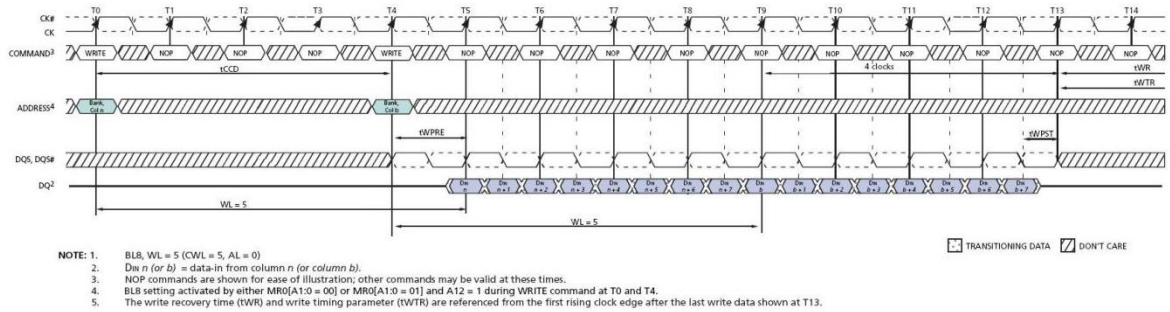


Figure 52 — WRITE (BL8) to WRITE (BL8)

图 13-2 DDR3 SDRAM 写操作协议

上图中，Cas Latency（CL） = 5， Wead Latency（WL）=5， Burst Length = 8。

DDR4 SDRAM 写操作协议类似。在图中命令 CMD 由 ACT_n、RAS_n、CAS_n 和 WE_n 共 4 个信号组成。对于读操作，ACT_n=1，RAS_n=1，CAS_n=0，WE_n=0。

13.4 DDR3/4 SDRAM 参数配置格式

13.4.1 内存控制器的参数列表

表 13-2 内存控制器软件可见参数列表

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
PHY								
0x0000							version(RD)	
0x0008			x4_mode	ddr3_mode			capability（RD）	
0x0010							dram_init(RD)	init_start
0x0018								
0x0020							preamble2	rdfifo_valid
0x0028		rdfifo_empty(RD)				Overflow(RD)		
0x0030		dll_value(RD)	dll_init_done(RD)	dll_lock_mode	dll_bypass	dll_adjj_cnt	dll_increment	dll_start_point
0x0038				dll_dbl_fix			dll_close_disable	dll_ck
0x0040				dbl_ctrl_ckca				dll_dbl_ckca
0x0048	pll_ctrl_ckca				pll_lock_ckca(RD)	dll_lock_ckca(RD)	clken_ckca	clksel_ckca
0x0050				dbl_ctrl_ds_0				dll_dbl_ds_0
0x0058	pll_ctrl_ds_0				pll_lock_ds_0(RD)	dll_lock_ds_0(RD)	clken_ds_0	clksel_ds_0
0x0060				dbl_ctrl_ds_1				dll_dbl_ds_1
0x0068	pll_ctrl_ds_1				pll_lock_ds_1(RD)	dll_lock_ds_1(RD)	clken_ds_1	clksel_ds_1
0x0070				dbl_ctrl_ds_2				dll_dbl_ds_2
0x0078	pll_ctrl_ds_2				pll_lock_ds_2(RD)	dll_lock_ds_2(RD)	clken_ds_2	clksel_ds_2
0x0080				dbl_ctrl_ds_3				dll_dbl_ds_3
0x0088	pll_ctrl_ds_3				pll_lock_ds_3(RD)	dll_lock_ds_3(RD)	clken_ds_3	clksel_ds_3
0x0090				dbl_ctrl_ds_4				dll_dbl_ds_4
0x0098	pll_ctrl_ds_4				pll_lock_ds_4(RD)	dll_lock_ds_4(RD)	clken_ds_4	clksel_ds_4
0x00a0				dbl_ctrl_ds_5				dll_dbl_ds_5
0x00a8	pll_ctrl_ds_5				pll_lock_ds_5(RD)	dll_lock_ds_5(RD)	clken_ds_5	clksel_ds_5
0x00b0				dbl_ctrl_ds_6				dll_dbl_ds_6
0x00b8	pll_ctrl_ds_6				pll_lock_ds_6(RD)	dll_lock_ds_6(RD)	clken_ds_6	clksel_ds_6
0x00c0				dbl_ctrl_ds_7				dll_dbl_ds_7
0x00c8	pll_ctrl_ds_7				pll_lock_ds_7(RD)	dll_lock_ds_7(RD)	clken_ds_7	clksel_ds_7
0x00d0				dbl_ctrl_ds_8				dll_dbl_ds_8
0x00d8	pll_ctrl_ds_8				pll_lock_ds_8(RD)	dll_lock_ds_8(RD)	clken_ds_8	clksel_ds_8
0x00e0			vrefclk_inv	vref_sample		vref_num	vref_dly	dll_vref
……								
0x0100					dll_1xdly_0	dll_1xgen_0	dll_wrdqs_0	dll_wrdq_0
0x0108						dll_gate_0	dll_rddqs1_0	dll_rddqs0_0
0x0110	rdodt_ctrl_0	rdgate_len_0	rdgate_mode_0	rdgate_ctrl_0			dqs_oe_ctrl_0	dq_oe_ctrl_0
0x0118						dly_2x_0	redge_sel_0	rddqs_phase_0(RD)
0x0120	w_bdly0_0[31:28]	w_bdly0_0[27:24]	w_bdly0_0[23:20]	w_bdly0_0[19:16]	w_bdly0_0[15:12]	w_bdly0_0[11:8]	w_bdly0_0[7:4]	w_bdly0_0[3:0]
0x0128		w_bdly0_0[59:56]	w_bdly0_0[55:52]	w_bdly0_0[51:48]	w_bdly0_0[47:44]	w_bdly0_0[43:40]	w_bdly0_0[39:36]	w_bdly0_0[35:32]
0x0130	w_bdly1_0[24:21]	w_bdly1_0[20:18]	w_bdly1_0[17:15]	w_bdly1_0[14:12]	w_bdly1_0[11:9]	w_bdly1_0[8:6]	w_bdly1_0[5:3]	w_bdly1_0[2:0]
0x0138								w_bdly1_0[27:26]
0x0140							rg_bdly_0[7:4]	rg_bdly_0[3:0]
0x0148								

0x0150	rdqsp_bdly_0[31:28]	rdqsp_bdly_0[27:24]	rdqsp_bdly_0[23:20]	rdqsp_bdly_0[19:16]	rdqsp_bdly_0[15:12]	rdqsp_bdly_0[11:8]	rdqsp_bdly_0[7:4]	rdqsp_bdly_0[3:0]
0x0158								rdqsp_bdly_0[35:32]
0x0160	rdqsn_bdly_0[31:28]	rdqsn_bdly_0[27:24]	rdqsn_bdly_0[23:20]	rdqsn_bdly_0[19:16]	rdqsn_bdly_0[15:12]	rdqsn_bdly_0[11:8]	rdqsn_bdly_0[7:4]	rdqsn_bdly_0[3:0]
0x0168								rdqsn_bdly_0[35:32]
0x0170	rdq_bdly_0[24:21]	rdq_bdly_0[20:18]	rdq_bdly_0[17:15]	rdq_bdly_0[14:12]	rdq_bdly_0[11:9]	rdq_bdly_0[8:6]	rdq_bdly_0[5:3]	rdq_bdly_0[2:0]
0x0178								rdq_bdly_0[27:26]
0x0180					dll_1xdly_1	dll_1xgen_1	dll_wrdqs_1	dll_wrdq_1
0x0188						dll_gate_1	dll_rddqs1_1	dll_rddqs0_1
0x0190	rdodt_ctrl_1	rdgate_len_1	rdgate_mode_1	rdgate_ctrl_1			dqs_oe_ctrl_1	dq_oe_ctrl_1
0x0198						dly_2x_1	redge_sel_1	rddqs_phase_1(RD)
0x01a0	w_bdly0_1[31:28]	w_bdly0_1[27:24]	w_bdly0_1[23:20]	w_bdly0_1[19:16]	w_bdly0_1[15:12]	w_bdly0_1[11:8]	w_bdly0_1[7:4]	w_bdly0_1[3:0]
0x01a8		w_bdly0_1[59:56]	w_bdly0_1[55:52]	w_bdly0_1[51:48]	w_bdly0_1[47:44]	w_bdly0_1[43:40]	w_bdly0_1[39:36]	w_bdly0_1[35:32]
0x01b0	w_bdly1_1[24:21]	w_bdly1_1[20:18]	w_bdly1_1[17:15]	w_bdly1_1[14:12]	w_bdly1_1[11:9]	w_bdly1_1[8:6]	w_bdly1_1[5:3]	w_bdly1_1[2:0]
0x01b8								w_bdly1_1[27:26]
0x01c0							rg_bdly_1[7:4]	rg_bdly_1[3:0]
0x01c8								
0x01d0	rdqsp_bdly_1[31:28]	rdqsp_bdly_1[27:24]	rdqsp_bdly_1[23:20]	rdqsp_bdly_1[19:16]	rdqsp_bdly_1[15:12]	rdqsp_bdly_1[11:8]	rdqsp_bdly_1[7:4]	rdqsp_bdly_1[3:0]
0x01d8								rdqsp_bdly_1[35:32]
0x01e0	rdqsn_bdly_1[31:28]	rdqsn_bdly_1[27:24]	rdqsn_bdly_1[23:20]	rdqsn_bdly_1[19:16]	rdqsn_bdly_1[15:12]	rdqsn_bdly_1[11:8]	rdqsn_bdly_1[7:4]	rdqsn_bdly_1[3:0]
0x01e8								rdqsn_bdly_1[35:32]
0x01f0	rdq_bdly_1[24:21]	rdq_bdly_1[20:18]	rdq_bdly_1[17:15]	rdq_bdly_1[14:12]	rdq_bdly_1[11:9]	rdq_bdly_1[8:6]	rdq_bdly_1[5:3]	rdq_bdly_1[2:0]
0x01f8								rdq_bdly_1[27:26]
0x0200					dll_1xdly_2	dll_1xgen_2	dll_wrdqs_2	dll_wrdq_2
0x0208						dll_gate_2	dll_rddqs1_2	dll_rddqs0_2
0x0210	rdodt_ctrl_2	rdgate_len_2	rdgate_mode_2	rdgate_ctrl_2			dqs_oe_ctrl_2	dq_oe_ctrl_2
0x0218						dly_2x_2	redge_sel_2	rddqs_phase_2(RD)
0x0220	w_bdly0_2[31:28]	w_bdly0_2[27:24]	w_bdly0_2[23:20]	w_bdly0_2[19:16]	w_bdly0_2[15:12]	w_bdly0_2[11:8]	w_bdly0_2[7:4]	w_bdly0_2[3:0]
0x0228		w_bdly0_2[59:56]	w_bdly0_2[55:52]	w_bdly0_2[51:48]	w_bdly0_2[47:44]	w_bdly0_2[43:40]	w_bdly0_2[39:36]	w_bdly0_2[35:32]
0x0230	w_bdly1_2[24:21]	w_bdly1_2[20:18]	w_bdly1_2[17:15]	w_bdly1_2[14:12]	w_bdly1_2[11:9]	w_bdly1_2[8:6]	w_bdly1_2[5:3]	w_bdly1_2[2:0]
0x0238								w_bdly1_2[27:26]
0x0240							rg_bdly_2[7:4]	rg_bdly_2[3:0]
0x0248								
0x0250	rdqsp_bdly_2[31:28]	rdqsp_bdly_2[27:24]	rdqsp_bdly_2[23:20]	rdqsp_bdly_2[19:16]	rdqsp_bdly_2[15:12]	rdqsp_bdly_2[11:8]	rdqsp_bdly_2[7:4]	rdqsp_bdly_2[3:0]
0x0258								rdqsp_bdly_2[35:32]
0x0260	rdqsn_bdly_2[31:28]	rdqsn_bdly_2[27:24]	rdqsn_bdly_2[23:20]	rdqsn_bdly_2[19:16]	rdqsn_bdly_2[15:12]	rdqsn_bdly_2[11:8]	rdqsn_bdly_2[7:4]	rdqsn_bdly_2[3:0]
0x0268								rdqsn_bdly_2[35:32]
0x0270	rdq_bdly_2[24:21]	rdq_bdly_2[20:18]	rdq_bdly_2[17:15]	rdq_bdly_2[14:12]	rdq_bdly_2[11:9]	rdq_bdly_2[8:6]	rdq_bdly_2[5:3]	rdq_bdly_2[2:0]

0x0278								rdq_bdly_2[27:26]
0x0280					dll_1xdly_3	dll_1xgen_3	dll_wrdqs_3	dll_wrdq_3
0x0288						dll_gate_3	dll_rddqs1_3	dll_rddqs0_3
0x0290	rdodt_ctrl_3	rdgate_len_3	rdgate_mode_3	rdgate_ctrl_3			dqs_oe_ctrl_3	dq_oe_ctrl_3
0x0298						dly_2x_3	redge_sel_3	rddqs_phase_3(RD)
0x02a0	w_bdly0_3[31:28]	w_bdly0_3[27:24]	w_bdly0_3[23:20]	w_bdly0_3[19:16]	w_bdly0_3[15:12]	w_bdly0_3[11:8]	w_bdly0_3[7:4]	w_bdly0_3[3:0]
0x02a8		w_bdly0_3[59:56]	w_bdly0_3[55:52]	w_bdly0_3[51:48]	w_bdly0_3[47:44]	w_bdly0_3[43:40]	w_bdly0_3[39:36]	w_bdly0_3[35:32]
0x02b0	w_bdly1_3[24:21]	w_bdly1_3[20:18]	w_bdly1_3[17:15]	w_bdly1_3[14:12]	w_bdly1_3[11:9]	w_bdly1_3[8:6]	w_bdly1_3[5:3]	w_bdly1_3[2:0]
0x02b8								w_bdly1_3[27:26]
0x02c0							rg_bdly_3[7:4]	rg_bdly_3[3:0]
0x02c8								
0x02d0	rdqsp_bdly_3[31:28]	rdqsp_bdly_3[27:24]	rdqsp_bdly_3[23:20]	rdqsp_bdly_3[19:16]	rdqsp_bdly_3[15:12]	rdqsp_bdly_3[11:8]	rdqsp_bdly_3[7:4]	rdqsp_bdly_3[3:0]
0x02d8								rdqsp_bdly_3[35:32]
0x02e0	rdqsn_bdly_3[31:28]	rdqsn_bdly_3[27:24]	rdqsn_bdly_3[23:20]	rdqsn_bdly_3[19:16]	rdqsn_bdly_3[15:12]	rdqsn_bdly_3[11:8]	rdqsn_bdly_3[7:4]	rdqsn_bdly_3[3:0]
0x02e8								rdqsn_bdly_3[35:32]
0x02f0	rdq_bdly_3[24:21]	rdq_bdly_3[20:18]	rdq_bdly_3[17:15]	rdq_bdly_3[14:12]	rdq_bdly_3[11:9]	rdq_bdly_3[8:6]	rdq_bdly_3[5:3]	rdq_bdly_3[2:0]
0x02f8								rdq_bdly_3[27:26]
0x0300					dll_1xdly_4	dll_1xgen_4	dll_wrdqs_4	dll_wrdq_4
0x0308						dll_gate_4	dll_rddqs1_4	dll_rddqs0_4
0x0310	rdodt_ctrl_4	rdgate_len_4	rdgate_mode_4	rdgate_ctrl_4			dqs_oe_ctrl_4	dq_oe_ctrl_4
0x0318						dly_2x_4	redge_sel_4	rddqs_phase_4(RD)
0x0320	w_bdly0_4[31:28]	w_bdly0_4[27:24]	w_bdly0_4[23:20]	w_bdly0_4[19:16]	w_bdly0_4[15:12]	w_bdly0_4[11:8]	w_bdly0_4[7:4]	w_bdly0_4[3:0]
0x0328		w_bdly0_4[59:56]	w_bdly0_4[55:52]	w_bdly0_4[51:48]	w_bdly0_4[47:44]	w_bdly0_4[43:40]	w_bdly0_4[39:36]	w_bdly0_4[35:32]
0x0330	w_bdly1_4[24:21]	w_bdly1_4[20:18]	w_bdly1_4[17:15]	w_bdly1_4[14:12]	w_bdly1_4[11:9]	w_bdly1_4[8:6]	w_bdly1_4[5:3]	w_bdly1_4[2:0]
0x0338								w_bdly1_4[27:26]
0x0340							rg_bdly_4[7:4]	rg_bdly_4[3:0]
0x0348								
0x0350	rdqsp_bdly_4[31:28]	rdqsp_bdly_4[27:24]	rdqsp_bdly_4[23:20]	rdqsp_bdly_4[19:16]	rdqsp_bdly_4[15:12]	rdqsp_bdly_4[11:8]	rdqsp_bdly_4[7:4]	rdqsp_bdly_4[3:0]
0x0358								rdqsp_bdly_4[35:32]
0x0360	rdqsn_bdly_4[31:28]	rdqsn_bdly_4[27:24]	rdqsn_bdly_4[23:20]	rdqsn_bdly_4[19:16]	rdqsn_bdly_4[15:12]	rdqsn_bdly_4[11:8]	rdqsn_bdly_4[7:4]	rdqsn_bdly_4[3:0]
0x0368								rdqsn_bdly_4[35:32]
0x0370	rdq_bdly_4[24:21]	rdq_bdly_4[20:18]	rdq_bdly_4[17:15]	rdq_bdly_4[14:12]	rdq_bdly_4[11:9]	rdq_bdly_4[8:6]	rdq_bdly_4[5:3]	rdq_bdly_4[2:0]
0x0378								rdq_bdly_4[27:26]
0x0380					dll_1xdly_5	dll_1xgen_5	dll_wrdqs_5	dll_wrdq_5
0x0388						dll_gate_5	dll_rddqs1_5	dll_rddqs0_5
0x0390	rdodt_ctrl_5	rdgate_len_5	rdgate_mode_5	rdgate_ctrl_5			dqs_oe_ctrl_5	dq_oe_ctrl_5
0x0398						dly_2x_5	redge_sel_5	rddqs_phase_5(RD)
0x03a0	w_bdly0_5[31:28]	w_bdly0_5[27:24]	w_bdly0_5[23:20]	w_bdly0_5[19:16]	w_bdly0_5[15:12]	w_bdly0_5[11:8]	w_bdly0_5[7:4]	w_bdly0_5[3:0]
0x03a8		w_bdly0_5[59:56]	w_bdly0_5[55:52]	w_bdly0_5[51:48]	w_bdly0_5[47:44]	w_bdly0_5[43:40]	w_bdly0_5[39:36]	w_bdly0_5[35:32]

0x03b0	w_bdly1_5[24:21]	w_bdly1_5[20:18]	w_bdly1_5[17:15]	w_bdly1_5[14:12]	w_bdly1_5[11:9]	w_bdly1_5[8:6]	w_bdly1_5[5:3]	w_bdly1_5[2:0]
0x03b8								w_bdly1_5[27:26]
0x03c0							rg_bdly_5[7:4]	rg_bdly_5[3:0]
0x03c8								
0x03d0	rdqsp_bdly_5[31:28]	rdqsp_bdly_5[27:24]	rdqsp_bdly_5[23:20]	rdqsp_bdly_5[19:16]	rdqsp_bdly_5[15:12]	rdqsp_bdly_5[11:8]	rdqsp_bdly_5[7:4]	rdqsp_bdly_5[3:0]
0x03d8								rdqsp_bdly_5[35:32]
0x03e0	rdqsn_bdly_5[31:28]	rdqsn_bdly_5[27:24]	rdqsn_bdly_5[23:20]	rdqsn_bdly_5[19:16]	rdqsn_bdly_5[15:12]	rdqsn_bdly_5[11:8]	rdqsn_bdly_5[7:4]	rdqsn_bdly_5[3:0]
0x03e8								rdqsn_bdly_5[35:32]
0x03f0	rdq_bdly_5[24:21]	rdq_bdly_5[20:18]	rdq_bdly_5[17:15]	rdq_bdly_5[14:12]	rdq_bdly_5[11:9]	rdq_bdly_5[8:6]	rdq_bdly_5[5:3]	rdq_bdly_5[2:0]
0x03f8								rdq_bdly_5[27:26]
0x0400					dll_1xdly_6	dll_1xgen_6	dll_wrdqs_6	dll_wrdq_6
0x0408						dll_gate_6	dll_rddqs1_6	dll_rddqs0_6
0x0410	rdodt_ctrl_6	rdgate_len_6	rdgate_mode_6	rdgate_ctrl_6			dqs_oe_ctrl_6	dq_oe_ctrl_6
0x0418						dly_2x_6	redge_sel_6	rddqs_phase_6(RD)
0x0420	w_bdly0_6[31:28]	w_bdly0_6[27:24]	w_bdly0_6[23:20]	w_bdly0_6[19:16]	w_bdly0_6[15:12]	w_bdly0_6[11:8]	w_bdly0_6[7:4]	w_bdly0_6[3:0]
0x0428		w_bdly0_6[59:56]	w_bdly0_6[55:52]	w_bdly0_6[51:48]	w_bdly0_6[47:44]	w_bdly0_6[43:40]	w_bdly0_6[39:36]	w_bdly0_6[35:32]
0x0430	w_bdly1_6[24:21]	w_bdly1_6[20:18]	w_bdly1_6[17:15]	w_bdly1_6[14:12]	w_bdly1_6[11:9]	w_bdly1_6[8:6]	w_bdly1_6[5:3]	w_bdly1_6[2:0]
0x0438								w_bdly1_6[27:26]
0x0440							rg_bdly_6[7:4]	rg_bdly_6[3:0]
0x0448								
0x0450	rdqsp_bdly_6[31:28]	rdqsp_bdly_6[27:24]	rdqsp_bdly_6[23:20]	rdqsp_bdly_6[19:16]	rdqsp_bdly_6[15:12]	rdqsp_bdly_6[11:8]	rdqsp_bdly_6[7:4]	rdqsp_bdly_6[3:0]
0x0458								rdqsp_bdly_6[35:32]
0x0460	rdqsn_bdly_6[31:28]	rdqsn_bdly_6[27:24]	rdqsn_bdly_6[23:20]	rdqsn_bdly_6[19:16]	rdqsn_bdly_6[15:12]	rdqsn_bdly_6[11:8]	rdqsn_bdly_6[7:4]	rdqsn_bdly_6[3:0]
0x0468								rdqsn_bdly_6[35:32]
0x0470	rdq_bdly_6[24:21]	rdq_bdly_6[20:18]	rdq_bdly_6[17:15]	rdq_bdly_6[14:12]	rdq_bdly_6[11:9]	rdq_bdly_6[8:6]	rdq_bdly_6[5:3]	rdq_bdly_6[2:0]
0x0478								rdq_bdly_6[27:26]
0x0480					dll_1xdly_7	dll_1xgen_7	dll_wrdqs_7	dll_wrdq_7
0x0488						dll_gate_7	dll_rddqs1_7	dll_rddqs0_7
0x0490	rdodt_ctrl_7	rdgate_len_7	rdgate_mode_7	rdgate_ctrl_7			dqs_oe_ctrl_7	dq_oe_ctrl_7
0x0498						dly_2x_7	redge_sel_7	rddqs_phase_7(RD)
0x04a0	w_bdly0_7[31:28]	w_bdly0_7[27:24]	w_bdly0_7[23:20]	w_bdly0_7[19:16]	w_bdly0_7[15:12]	w_bdly0_7[11:8]	w_bdly0_7[7:4]	w_bdly0_7[3:0]
0x04a8		w_bdly0_7[59:56]	w_bdly0_7[55:52]	w_bdly0_7[51:48]	w_bdly0_7[47:44]	w_bdly0_7[43:40]	w_bdly0_7[39:36]	w_bdly0_7[35:32]
0x04b0	w_bdly1_7[24:21]	w_bdly1_7[20:18]	w_bdly1_7[17:15]	w_bdly1_7[14:12]	w_bdly1_7[11:9]	w_bdly1_7[8:6]	w_bdly1_7[5:3]	w_bdly1_7[2:0]
0x04b8								w_bdly1_7[27:26]
0x04c0							rg_bdly_7[7:4]	rg_bdly_7[3:0]
0x04c8								
0x04d0	rdqsp_bdly_7[31:28]	rdqsp_bdly_7[27:24]	rdqsp_bdly_7[23:20]	rdqsp_bdly_7[19:16]	rdqsp_bdly_7[15:12]	rdqsp_bdly_7[11:8]	rdqsp_bdly_7[7:4]	rdqsp_bdly_7[3:0]
0x04d8								rdqsp_bdly_7[35:32]

0x04e0	rdqsn_bdly_7[31:28]	rdqsn_bdly_7[27:24]	rdqsn_bdly_7[23:20]	rdqsn_bdly_7[19:16]	rdqsn_bdly_7[15:12]	rdqsn_bdly_7[11:8]	rdqsn_bdly_7[7:4]	rdqsn_bdly_7[3:0]
0x04e8								rdqsn_bdly_7[35:32]
0x04f0	rdq_bdly_7[24:21]	rdq_bdly_7[20:18]	rdq_bdly_7[17:15]	rdq_bdly_7[14:12]	rdq_bdly_7[11:9]	rdq_bdly_7[8:6]	rdq_bdly_7[5:3]	rdq_bdly_7[2:0]
0x04f8								rdq_bdly_7[27:26]
0x0500					dll_1xdly_8	dll_1xgen_8	dll_wrdqs_8	dll_wrdq_8
0x0508						dll_gate_8	dll_rddqs1_8	dll_rddqs0_8
0x0510	rdodt_ctrl_8	rdgate_len_8	rdgate_mode_8	rdgate_ctrl_8			dqs_oe_ctrl_8	dq_oe_ctrl_8
0x0518						dly_2x_8	redge_sel_8	rddqs_phase_8(RD)
0x0520	w_bdly0_8[31:28]	w_bdly0_8[27:24]	w_bdly0_8[23:20]	w_bdly0_8[19:16]	w_bdly0_8[15:12]	w_bdly0_8[11:8]	w_bdly0_8[7:4]	w_bdly0_8[3:0]
0x0528		w_bdly0_8[59:56]	w_bdly0_8[55:52]	w_bdly0_8[51:48]	w_bdly0_8[47:44]	w_bdly0_8[43:40]	w_bdly0_8[39:36]	w_bdly0_8[35:32]
0x0530	w_bdly1_8[24:21]	w_bdly1_8[20:18]	w_bdly1_8[17:15]	w_bdly1_8[14:12]	w_bdly1_8[11:9]	w_bdly1_8[8:6]	w_bdly1_8[5:3]	w_bdly1_8[2:0]
0x0538								w_bdly1_8[27:26]
0x0540							rg_bdly_8[7:4]	rg_bdly_8[3:0]
0x0548								
0x0550	rdqsp_bdly_8[31:28]	rdqsp_bdly_8[27:24]	rdqsp_bdly_8[23:20]	rdqsp_bdly_8[19:16]	rdqsp_bdly_8[15:12]	rdqsp_bdly_8[11:8]	rdqsp_bdly_8[7:4]	rdqsp_bdly_8[3:0]
0x0558								rdqsp_bdly_8[35:32]
0x0560	rdqsn_bdly_8[31:28]	rdqsn_bdly_8[27:24]	rdqsn_bdly_8[23:20]	rdqsn_bdly_8[19:16]	rdqsn_bdly_8[15:12]	rdqsn_bdly_8[11:8]	rdqsn_bdly_8[7:4]	rdqsn_bdly_8[3:0]
0x0568								rdqsn_bdly_8[35:32]
0x0570	rdq_bdly_8[24:21]	rdq_bdly_8[20:18]	rdq_bdly_8[17:15]	rdq_bdly_8[14:12]	rdq_bdly_8[11:9]	rdq_bdly_8[8:6]	rdq_bdly_8[5:3]	rdq_bdly_8[2:0]
0x0578								rdq_bdly_8[27:26]
……								
0x0700					leveling_cs	tLVL_DELAY	leveling_req(WR)	leveling_mode
0x0708							leveling_done(RD)	leveling_ready(RD)
0x0710	leveling_resp_7	leveling_resp_6	leveling_resp_5	leveling_resp_4	leveling_resp_3	leveling_resp_2	leveling_resp_1	leveling_resp_0
0x0718								leveling_resp_8
0x0720								
……								
0x0800	dfe_ctrl_ds	pad_ctrl_ds				pad_ctrl_ck		
0x0808		pad_reset_po	pad_oplen_ca	pad_opdly_ca		pad_ctrl_ca		
0x0810	vref_ctrl_ds_3		vref_ctrl_ds_2		vref_ctrl_ds_1		vref_ctrl_ds_0	
0x0818	vref_ctrl_ds_7		vref_ctrl_ds_6		vref_ctrl_ds_5		vref_ctrl_ds_4	
0x0820							vref_ctrl_ds_8	
0x0828								
0x0830			pad_comp_o(RD)				pad_comp_i	
0x0838								
CTL								
0x1000		tRP	tWLDQSEN	tMOD	tXPR		tCKE	tRESET
0x1008								tODTL
0x1010	tREFretention				tRFC		tREF	
0x1018	tCKESR	tXSRD	tXS		tRFC_dlr			tREF_IDLE

0x1020					tRDPDEN	tCPDED	tXPDLL	tXP
0x1028					tZQperiod	tZQCL	tZQCS	tZQ_CMD
……								
0x1040	tRCD	tRRD_S_slr	tRRD_L_slr	tRRD_dlr				tRAS_min
0x1048				tRTP	tWR_CRC_DM	tWR	tFAW_slr	tFAW
0x1050	tWTR_S_CRC_DM	tWTR_L_CRC_DM	tWTR_S	tWTR		tCCD_dlr	tCCD_S_slr	tCCD_L_slr
0x1058								
0x1060			tPHY_WRLAT	tWL		tRDDATA	tPHY_RDLAT	tRL
0x1068				tCAL				tPL
0x1070			tW2P_sameba	tW2W_sameba	tW2R_sameba	tR2P_sameba	tR2W_sameba	tR2R_sameba
0x1078			tW2P_samebg	tW2W_samebg	tW2R_samebg	tR2P_samebg	tR2W_samebg	tR2R_samebg
0x1080			tW2P_samec	tW2W_samec	tW2R_samec	tR2P_samec	tR2W_samec	tR2R_samec
0x1088								
0x1090			tW2P_samecs	tW2W_samecs	tW2R_samecs	tR2P_samecs	tR2W_samecs	tR2R_samecs
0x1098				tW2W_diffcs	tW2R_diffcs		tR2W_diffcs	tR2R_diffcs
……								
0x1100			cs_ref	cs_resync	cs_zqcl	cs_zq	cs_mrs	cs_enable
0x1108	cke_map				cs_map			
0x1110				cs2cid				cid_map
0x1118								
0x1120	mrs_done(RD)	mrs_req(WR)	pre_all_done(RD)	pre_all_req(WR)	cmd_cmd	status_cmd(RD)	cmd_req(WR)	command_mode
0x1128	cmd_cke	cmd_a			cmd_ba	cmd_bg	cmd_c	cmd_cs
0x1130								cmd_pda
0x1138						cmd_dq0		
0x1140	mr_3_cs_0		mr_2_cs_0		mr_1_cs_0		mr_0_cs_0	
0x1148	mr_3_cs_1		mr_2_cs_1		mr_1_cs_1		mr_0_cs_1	
0x1150	mr_3_cs_2		mr_2_cs_2		mr_1_cs_2		mr_0_cs_2	
0x1158	mr_3_cs_3		mr_2_cs_3		mr_1_cs_3		mr_0_cs_3	
0x1160	mr_3_cs_4		mr_2_cs_4		mr_1_cs_4		mr_0_cs_4	
0x1168	mr_3_cs_5		mr_2_cs_5		mr_1_cs_5		mr_0_cs_5	
0x1170	mr_3_cs_6		mr_2_cs_6		mr_1_cs_6		mr_0_cs_6	
0x1178	mr_3_cs_7		mr_2_cs_7		mr_1_cs_7		mr_0_cs_7	
0x1180	mr_3_cs_0_ddr4		mr_2_cs_0_ddr4		mr_1_cs_0_ddr4		mr_0_cs_0_ddr4	
0x1188			mr_6_cs_0_ddr4		mr_5_cs_0_ddr4		mr_4_cs_0_ddr4	
0x1190	mr_3_cs_1_ddr4		mr_2_cs_1_ddr4		mr_1_cs_1_ddr4		mr_0_cs_1_ddr4	
0x1198			mr_6_cs_1_ddr4		mr_5_cs_1_ddr4		mr_4_cs_1_ddr4	
0x11a0	mr_3_cs_2_ddr4		mr_2_cs_2_ddr4		mr_1_cs_2_ddr4		mr_0_cs_2_ddr4	
0x11a8			mr_6_cs_2_ddr4		mr_5_cs_2_ddr4		mr_4_cs_2_ddr4	
0x11b0	mr_3_cs_3_ddr4		mr_2_cs_3_ddr4		mr_1_cs_3_ddr4		mr_0_cs_3_ddr4	
0x11b8			mr_6_cs_3_ddr4		mr_5_cs_3_ddr4		mr_4_cs_3_ddr4	
0x11c0	mr_3_cs_4_ddr4		mr_2_cs_4_ddr4		mr_1_cs_4_ddr4		mr_0_cs_4_ddr4	
0x11c8			mr_6_cs_4_ddr4		mr_5_cs_4_ddr4		mr_4_cs_4_ddr4	
0x11d0	mr_3_cs_5_ddr4		mr_2_cs_5_ddr4		mr_1_cs_5_ddr4		mr_0_cs_5_ddr4	

0x11d8			mr_6_cs_5_ddr4		mr_5_cs_5_ddr4		mr_4_cs_5_ddr4	
0x11e0	mr_3_cs_6_ddr4		mr_2_cs_6_ddr4		mr_1_cs_6_ddr4		mr_0_cs_6_ddr4	
0x11e8			mr_6_cs_6_ddr4		mr_5_cs_6_ddr4		mr_4_cs_6_ddr4	
0x11f0	mr_3_cs_7_ddr4		mr_2_cs_7_ddr4		mr_1_cs_7_ddr4		mr_0_cs_7_ddr4	
0x11f8			mr_6_cs_7_ddr4		mr_5_cs_7_ddr4		mr_4_cs_7_ddr4	
0x1200			nc16_map	nc	channel_width	ba_xor_row_offset	addr_new	cs_place
0x1208						bg_xor_row_offset		addr_mirror
0x1210	addr_base_1				addr_base_0			
0x1218								
0x1220	addr_mask_1				addr_mask_0			
0x1228								
0x1230			cs_diff	c_diff	bg_diff	ba_diff	row_diff	col_diff
0x1238				CF_confbus_timeout				
0x1240	WRQthreshold	tRDQidle	wr_pkc_num	rwq_rb	retry	no_dead_inorder	placement_en	stb_en/pbuf
0x1248								tRWGNTidle
0x1250							rfifo_age	
0x1258	prior_age3		prior_age2		prior_age1		prior_age0	
0x1260	retry_cnt(RD)					rbuffer_max(RD)	rdfifo_depth	stat_en
0x1268								
……								
0x1280	aw_512_align		rd_before_wr	ecc_enable		int_vector(RD)	int_trigger(RD)	int_enable
0x1288								
0x1290						int_cnt_fatal(RD)	int_cnt_err(RD)	int_cnt
0x1298	ecc_cnt_cs_7(RD)	ecc_cnt_cs_6(RD)	ecc_cnt_cs_5(RD)	ecc_cnt_cs_4(RD)	ecc_cnt_cs_3(RD)	ecc_cnt_cs_2(RD)	ecc_cnt_cs_1(RD)	ecc_cnt_cs_0(RD)
0x12a0	ecc_data_dir(RD)	ecc_code_dir(RD)	ecc_code_256(RD)					ecc_code_64(RD)
0x12a8	ecc_addr(RD)							
0x12b0	ecc_data[63:0](RD)							
0x12b8	ecc_data[127:64] (RD)							
0x12c0	ecc_data[191:128] (RD)							
0x12c8	ecc_data[255:192] (RD)							
……								
0x1300							ref_num	ref_sch_en
0x1308							Status_sref(RD)	srefresh_req
……								
0x1340	hardware_pd_7	hardware_pd_6	hardware_pd_5	hardware_pd_4	hardware_pd_3	hardware_pd_2	hardware_pd_1	hardware_pd_0
0x1348	power_sta_7(RD)	power_sta_6(RD)	power_sta_5(RD)	power_sta_4(RD)	power_sta_3(RD)	power_sta_2(RD)	power_sta_1(RD)	power_sta_0(RD)
0x1350	selfref_age		slowpd_age		fastpd_age		active_age	
0x1358				power_up				Age_step
0x1360	tCONF_IDLE				tLPMC_IDLE			
……								
0x1380								zq_overlap
0x1388								zq_stat_en
0x1390	zq_cnt_1(RD)				zq_cnt_0(RD)			

0x1398	zq_cnt_3(RD)				zq_cnt_2(RD)			
0x13a0	zq_cnt_5(RD)				zq_cnt_4(RD)			
0x13a8	zq_cnt_6(RD)				zq_cnt_6(RD)			
……								
0x13c0					odt_wr_cs_map			
0x13c8							odt_wr_length	odt_wr_delay
0x13d0					odt_rd_cs_map			
0x13d8							odt_rd_length	odt_rd_delay
……								
0x1400				tRESYNC_length	tRESYNC_delay	tRESYNC_shift	tRESYNC_max	tRESYNC_min
……								
0x1440					pre_predict		tm_cmdq_num	burst_length
0x1448								ca_timing
0x1450						wr/rd_dbi_en	ca_par_en	crc_en
0x1458							tCA_PAR	tWR_CRC
0x1460	bit_map_7	bit_map_6	bit_map_5	bit_map_6	bit_map_3	bit_map_2	bit_map_1	bit_map_0
0x1468	bit_map_15	bit_map_14	bit_map_13	bit_map_12	bit_map_11	bit_map_10	bit_map_9	bit_map_8
0x1470							bit_map_17	bit_map_16
0x1478								bitmap_mirror
0x1480				alertn_misc(RD)			alertn_cnt	alertn_clr
0x1488	alertn_addr(RD)							
……								
0x1500	win0_base							
0x1508	win1_base							
0x1510	win2_base							
0x1518	win3_base							
0x1520	win4_base							
0x1528	win5_base							
0x1530	win6_base							
0x1538	win7_base							
……								
0x1580	win0_mask							
0x1588	win1_mask							
0x1590	win2_mask							
0x1598	win3_mask							
0x15a0	win4_mask							
0x15a8	win5_mask							
0x15b0	win6_mask							
0x15b8	win7_mask							
……								
0x1600	win0_mmap							
0x1608	win1_mmap							
0x1610	win2_mmap							

0x1618	win3_mmap							
0x1620	win4_mmap							
0x1628	win5_mmap							
0x1630	win6_mmap							
0x1638	win7_mmap							
……								
0x1700							acc_hp	acc_en
0x1708	acc_fake_b				acc_fake_a			
0x1710								
0x1718								
0x1720	addr_base_acc_1				addr_base_acc_0			
0x1728								
0x1730	addr_mask_acc_1				addr_mask_acc_0			
0x1738								
MON								
0x2000								cmd_monitor
0x2008								
0x2010	cmd_fbck[63:0](RD)							
0x2018	cmd_fbck[127:64] (RD)							
0x2020					rw_switch_cnt(RD)			
……								
0x2100								scheduler_mon
0x2108								
0x2110	sch_cmd_num(RD)							
0x2118	ba_conflict_all(RD)							
0x2120	ba_conflict_last1(RD)							
0x2128	ba_conflict_last2(RD)							
0x2130	ba_conflict_last3(RD)							
0x2138	ba_conflict_last4(RD)							
0x2140	ba_conflict_last5(RD)							
0x2148	ba_conflict_last6(RD)							
0x2150	ba_conflict_last7(RD)							
0x2158	ba_conflict_last8(RD)							
0x2160	rd_conflict(RD)							
0x2168	wr_conflict(RD)							
0x2170	rtw_conflict(RD)							
0x2178	wtr_conflict(RD)							
0x2180	rd_conflict_last1(RD)							
0x2188	wr_conflict_last1(RD)							
0x2190	rtw_conflict_last1(RD)							
0x2198	wtr_conflict_last1(RD)							
0x21a0	wr_rd_turnaround(RD)							
0x21a8	cs_turnaround(RD)							

0x21b0	bg_conflict(RD)							
……								
0x2300						sm_leveling		sm_init
0x2308								
0x2310		sm_rank_03		sm_rank_02		sm_rank_01		sm_rank_00
0x2318		sm_rank_07		sm_rank_06		sm_rank_05		sm_rank_04
0x2320		sm_rank_11		sm_rank_10		sm_rank_09		sm_rank_08
0x2328		sm_rank_15		sm_rank_14		sm_rank_13		sm_rank_12
0x2330		sm_rank_19		sm_rank_18		sm_rank_17		sm_rank_16
0x2338		sm_rank_23		sm_rank_22		sm_rank_21		sm_rank_20
0x2340		sm_rank_27		sm_rank_26		sm_rank_25		sm_rank_24
0x2348		sm_rank_31		sm_rank_30		sm_rank_29		sm_rank_28
……								
TST								
0x3000						lpbk_mode	lpbk_start	lpbk_en
0x3008	lpbk_correct(RD)				lpbk_counter(RD)			lpbk_error(RD)
0x3010	lpbk_data_en[63:0]							
0x3018								lpbk_data_en[71:64]
0x3020							lpbk_data_mask_en	
0x3028								
0x3030	Lpbk_dat_w0[63:0]							
0x3038	Lpbk_dat_w0[127:64]							
0x3040	Lpbk_dat_w1[63:0]							
0x3048	Lpbk_dat_w1[127:64]							
0x3050		lpbk_ecc_mask_w0	lpbk_dat_mask_w0				lpbk_ecc_w0	
0x3058		lpbk_ecc_mask_w1	lpbk_dat_mask_w1				lpbk_ecc_w1	
0x3060								prbs_23
0x3068						prbs_init		
……								
0x3100					fix_data_pattern_index	bus_width	page_size	test_engine_en
0x3108			cs_diff_tst	c_diff_tst	bg_diff_tst	ba_diff_tst	row_diff_tst	col_diff_tst
0x3120	addr_base_tst							
0x3128								
0x3130	user_data_pattern							
0x3138								
0x3140	valid_bits[63:0]							
0x3148								valid_bits[71:64]
0x3150	ctrl[63:0]							
0x3158	ctrl[127:64]							
0x3160	obs[63:0] (RD)							

0x3168	obs[127:64] (RD)							
0x3170	obs[191:128] (RD)							
0x3178	obs[255:192] (RD)							
0x3180	obs[319:256] (RD)							
0x3188	obs[383:320] (RD)							
0x3190	obs[447:384] (RD)							
0x3198	obs[511:448] (RD)							
0x31a0	obs[575:512] (RD)							
0x31a8	obs[639:576] (RD)							
0x31b0					obs[671:640](RD)			
……								
0x3200								
0x3208								
0x3220	tud_i0							
0x3228	tud_i1							
0x3230	tud_o(RD)							
……								
0x3300	tst_300							
0x3308	tst_308							
0x3310	tst_310							
0x3318	tst_318							
0x3320	tst_320							
0x3328	tst_328							
0x3330	tst_330							
0x3338	tst_338							
0x3340	tst_340							
0x3348	tst_348							
0x3350	tst_350							
0x3358	tst_358							
0x3360	tst_360							
0x3368	tst_368							
0x3370	tst_370							
0x3378	tst_378							

13.5 软件编程指南

13.5.1 初始化操作

初始化操作由软件向寄存器 Init_start（0x010）写入 0x2 时开始，在设置 Init_start 信号之前，必须将其它所有寄存器设置为正确的值。

软硬件协同的 DRAM 初始化过程如下：

(1) 设置 pm_clk_sel_ckca 和 pm_clk_sel_ds

(2) 设置 pm_phy_init_start 为 1，开始初始化 PHY

(3) 等待 DLL 主控模块锁定，即 pm_dll_init_done 为 1

(4) 等待所有时钟产生模块的 pm_dll_lock_*或者 pm_pll_lock_*变为 1

(5) 使能所有的 pm_clken_*

(6) 将 pm_init_start 设置为 1，内存控制器开始初始化

(7) 等待内存控制器初始化完成，即 pm_dram_init 的值与 pm_cs_enable 相同。

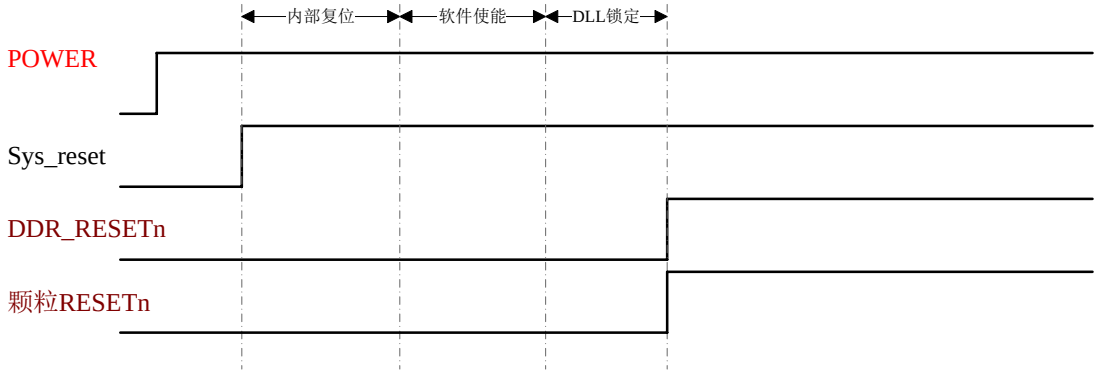
13.5.2 复位引脚的控制

为了在 STR 等状态下更加简单地控制复位引脚，可以通过 pad_reset_po（0x808）寄存器进行特别的复位引脚（DDR_RESETn）控制，主要的控制模式有两种：

(1) 一般模式，reset_ctrl[1:0] == 2’b00。这种模式下，复位信号引脚的行为与一般的控制模式相兼容。主板上直接将 DDR_RESETn 与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始初始化时，引脚状态为高；
- 正常工作时，引脚状态为高。

时序如下图所示：

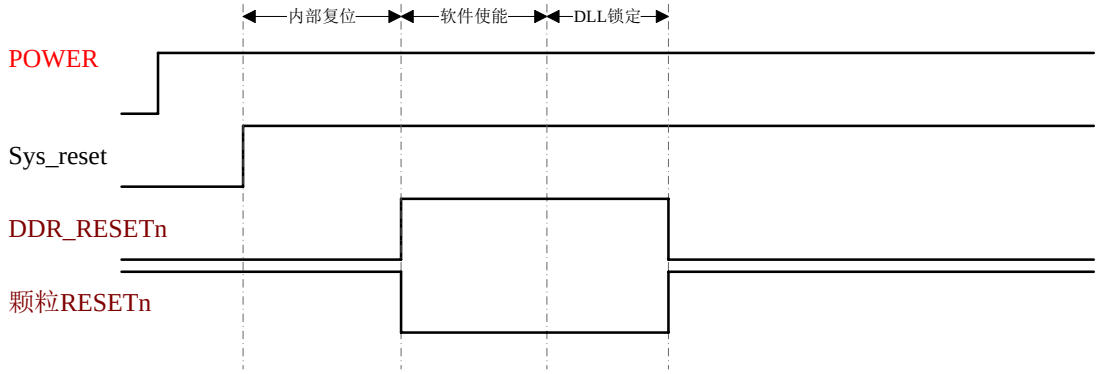


(2) 反向模式，reset_ctrl[1:0] == 2’b10。这种模式下，复位信号引脚在进行内存实际控制的时候，有效电平与一般的控制模式相反。所以主板上需要将 DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；

- 上电时：引脚状态为低；
- 控制器开始配置时：引脚状态为高；
- 控制器开始初始化时：引脚状态为低；
- 正常工作时：引脚状态为低。

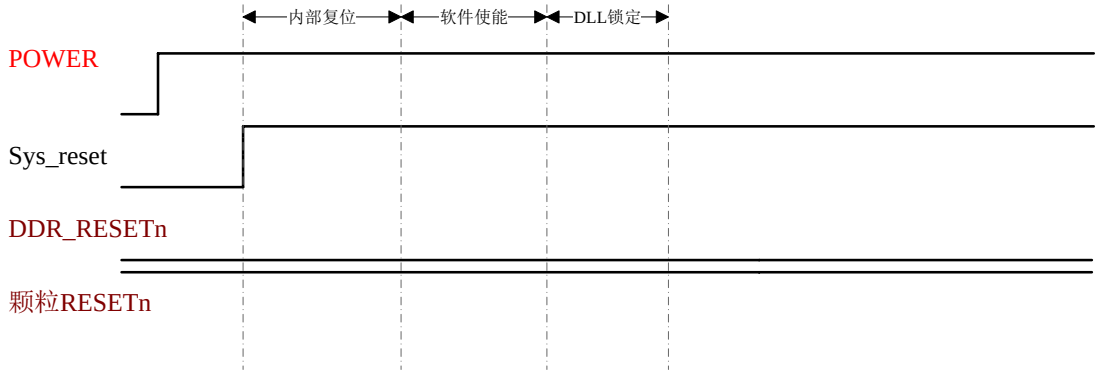
时序如下图所示：



(3) 复位禁止模式，pm_pad_reset_o[1:0] == 2’b01。这种模式下，复位信号引脚在整个内存工作期间，保持低电平。所以主板上需要将 DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 始终为低；

时序如下图所示：



由后两种复位模式相配合，就可以直接在使用内存控制器的复位信号的情况下实现 STR 控制。当整个系统从关闭状态下启动时，使用（2）中的方法来使用内存条正常复位并开始工作。当系统从 STR 中恢复的时候，使用（3）中的方法来重新配置内存条，使得在不破坏内存条原有状态的条件上使其重新开始正常工作。

13.5.3 Leveling

Leveling 操作是在 DDR3/4 中，用于智能配置内存控制器读写操作中各种信号间相位关

系的操作。通常它包括了 Write Leveling、Read Leveling 和 Gate Leveling。在本控制器中，只实现了 Write Leveling 与 Gate Leveling，Read Leveling 没有实现，软件需要通过判断读写的正确性来实现 Read Leveling 所完成的功能。除了在 Leveling 过程中操作的 DQS 相位、GATE 相位之外，还可以根据这些最后确认的相位来计算出写 DQ 相位、读 DQ 相位的配置方法。此外，本设计还支持 bit-deskew 功能，用于补偿一个 dataslice 内不同 bit 之间的延时差。

13.5.3.1 Write Leveling

Write Leveling 用于配置写 DQS 与时钟之间的相位关系,软件编程需要参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；
(2) 将 Dll_wrdqs_x（x = 0…8）设置为 0x20；
(3) 将 Dll_wrdq_x（x = 0…8）设置为 0x0；
(4) 设置 Lvl_mode 为 2’b01；
(5) 采样 Lvl_ready 寄存器，如果为 1，表示可以开始 Write Leveling 请求；
(6) 设置 Lvl_req 为 1；
(7) 采样 Lvl_done 寄存器，如果为 1，表示一次 Write Leveling 请求完成；
(8) 采样 Lvl_resp_x 寄存器，如果为 0，则将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9；如果为 1，则将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 0，然后继续将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9。
(9) 将 Dll_wrdq_x 和 dll_1xdly 的值减 0x40，此时 Dll_wrdq_x 和 dll_1xdly 的值就应该是正确的设置值。
(10) 根据 DIMM 类型设置 pm_dly_2x，对于 0x0 边界右边的颗粒对应的 pm_dly_2x 值增加 0x010101。
(11) 将 Lvl_mode（0x700）设置为 2’b00，退出 Write Leveling 模式。

13.5.3.2 Gate Leveling

Gate Leveling 用于配置控制器内使能采样读 DQS 窗口的时机，软件编程参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；
(2) 完成 Write Leveling，参见上一小节内容；
(3) 将 Dll_gate_x（x = 0…8）设置为 0；
(4) 设置 Lvl_mode 为 2’b10；
(5) 采样 Lvl_ready 寄存器，如果为 1，表示可以开始 Gate Leveling 请求；
(6) 设置 Lvl_req 为 1；
(7) 采样 Lvl_done 寄存器，如果为 1，表示一次 Gate Leveling 请求完成；

(8) 采样 Lvl_resp_x[0]寄存器。如果第一次采样发现 Lvl_resp_x[0]为 1，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6-8，直至采样结果为 0，否则进行下一步；

(9) 如果采样结果为 0，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6-9；如果为 1，则表示 Gate Leveling 操作已经成功；

(10) 根据 pm_rddqs_phase 的值设置 pm_rdedge_sel

(11) 将 Dll_gate_x（x = 0…8）减 0x20；

(12) 调整完毕后，再分别进行两次 Lvl_req 操作，观察 Lvl_resp_x[7:5]与 Lvl_resp_x[4:2]的值变化，如果各增加为 Burst_length/2，则继续进行第 13 步操作；如果不为 4，可能需要对 Rd_oe_begin_x 进行加一或减一操作，如果大于 Burst_length/2，很可能需要对 Dll_gate_x 的值进行一些微调；

(13) 将 Lvl_mode（0x700）设置为 2’b00，退出 Gate Leveling 模式；

(14) 至此 Gate Leveling 操作结束。

13.5.4 功耗控制配置流程

首先需要设置 pm_pad_ctrl_ca[0]为 1，等待内存初始化完成之后，再设置 pm_pad_ctrl_ca[0]为 0。该功能只有 DDR4 模式下使能了 CAL Mode 才可以使用。

13.5.5 单独发起 MRS 命令

DDR3 模式时，内存控制器向内存发出的 MRS 命令次序分别为：

MR2_CS0、MR2_CS1、MR2_CS2、MR2_CS3、MR2_CS4、MR2_CS5、MR2_CS6、MR2_CS7、

MR3_CS0、MR3_CS1、MR3_CS2、MR3_CS3、MR3_CS4、MR3_CS5、MR3_CS6、MR3_CS7、

MR1_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR1_CS4、MR1_CS5、MR1_CS6、MR1_CS7、

MR0_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR0_CS4、MR0_CS5、MR0_CS6、MR0_CS7。

另外，对于 DDR4 模式时，内存控制器向内存发出的 MRS 命令次序分别为：

MR3_CS0、MR3_CS1、MR3_CS2、MR3_CS3、MR3_CS4、MR3_CS5、MR3_CS6、MR3_CS7、

MR6_CS0、MR6_CS1、MR6_CS2、MR6_CS3、MR6_CS4、MR6_CS5、MR6_CS6、MR6_CS7、

MR5_CS0、MR5_CS1、MR5_CS2、MR5_CS3、MR5_CS4、MR5_CS5、MR5_CS6、MR5_CS7、

MR4_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR4_CS4、MR4_CS5、MR4_CS6、MR4_CS7、

MR2_CS0、MR2_CS1、MR2_CS2、MR2_CS3、MR2_CS4、MR2_CS5、MR2_CS6、MR2_CS7、

MR1_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR1_CS4、MR1_CS5、MR1_CS6、MR1_CS7、

MR0_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR0_CS4、MR0_CS5、MR0_CS6、MR0_CS7。

其中，对应 CS 的 MRS 命令是否有效，是由 Cs_mrs 决定，只有 Cs_mrs 上对应每个片选的位有效，才会真正向 DRAM 发出这个 MRS 命令。对应的每个 MR 的值由寄存器 Mr*_cs*决

定。这些值同时也用于初始化内存时的 MRS 命令。

具体操作如下：

(1) 将寄存器 Cs_mrs（0x1101）、Mr*_cs*（0x1140 – 0x11f8）设置为正确的值；
(2) 设置 Command_mode（0x0x1120）为 1，使控制器进入命令发送模式；
(3) 采样 Status_cmd（0x1122），如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；
(4) 写 Mrs_req（0x1126）为 1，向 DRAM 发送 MRS 命令；
(5) 采样 Mrs_done（0x1127），如果为 1，则表示 MRS 命令已经发送完毕，可以退出，如果为 0，则需要继续等待；
(6) 设置 Command_mode（0x1120）为 0，使控制器退出命令发送模式。

13.5.6 任意操作控制总线

内存控制器可以通过命令发送模式向 DRAM 发出任意的命令组合，软件可以设置 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a（0x1128），在命令发送模式下向 DRAM 发出。

具体操作如下：

(1) 将寄存器 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a（0x1128）设置为正确的值；
(2) 设置 Command_mode（0x1120）为 1，使控制器进入命令发送模式；
(3) 采样 Status_cmd（0x1122），如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；
(4) 写 Cmd_req（0x1121）为 1，向 DRAM 发送命令；
(5) 设置 Command_mode（0x1120）为 0，使控制器退出命令发送模式。

13.5.7 自循环测试模式控制

自循环测试模式可以分别在测试模式下或者正常功能模式下使用，为此，本内存控制器分别实现了两套独立的控制接口，一套用于在测试模式下由测试端口直接控制，另一套用于在正常功能模式下由寄存器配置模块进行配置使能测试。

这两套接口的复用使用端口 test_phy 进行控制，当 test_phy 有效时，使用控制器的 test_*端口进行控制，此时的自测试完全由硬件控制；当 test_phy 无效时，使用软件编程的 pm_*的参数进行控制。使用测试端口的具体信号含义可以参考寄存器参数中的同名部分。

这两套接口从控制的参数来说基本一致，仅仅是接入点不同，在此介绍软件编程时的控制方法。具体操作如下：

(1) 将内存控制器所有的参数全部正确设置；

(2) 按照初始化流程等待时钟复位稳定；

(3) 将寄存器 Lpbk_en 设为 1；

(4) 将寄存器 Lpbk_start 设为 1；此时自循环测试正式开始。

(5) 到此为止自循环测试已经开始，软件需要经常检测是否有错误发生，具体操作如下：

(6) 采样寄存器 Lpbk_error，如果这个值为 1，表示有错误发生，此时可以通过 Lpbk_*等观测用寄存器来观测第一个出错时的错误数据和正确数据；如果这个值为 0，表示还没有出现过数据错误。

13.5.8 ECC 功能使用控制

ECC 功能只有在 64 位模式下可以使用。

Ecc_enable（0x1280）包括以下 2 个控制位：

Ecc_enable[0]控制是否使能 ECC 功能，只有设置了这个有效位，才会使能 ECC 功能。

Ecc_enable[1]控制是否通过处理器内部的读响应通路进行报错，以使得出现 ECC 两位错的读访问能立即导致处理器核的异常发生。

除此之外，ECC 出错还可以通过中断方式通知处理器核。这个中断通过 Int_enable 进行控制。中断包括两个向量，Int_vector[0]表示出现 ECC 错误（包括 1 位错与 2 位错），Int_vecotr[1]表示出现 ECC 两位错。Int_vector 的清除通过向对应位写 1 实现。

13.5.9 出错状态观测

内存控制器出错后，可通过访问相应的系统配置寄存器来获取相应的出错信息，并进行简单的调试操作。寄存器基地址为 0x1fe00000 或 0x3ff00000，同样也可以使用配置寄存器指令进行访问，寄存器及其对应位如下。

表 13-3 0 号内存控制器出错状态观测寄存器

寄存器	偏移地址	控制	说明
0 号内存控制器 ECC 设置寄存器 Mc0_ecc_set	0x0600	RW	0 号内存控制器 ECC 设置寄存器 [5:0]：MC0 int_enable，中断使能 [8]：MC0 int_trigger，中断触发配置 [21:16]：MC0 int_vector(RO)，中断向量（只读） [33:32]：MC0 ecc_enable，ECC 相关功能使能 [40]：MC0 rd_before_wr，读后写功能使能
	0x0608	RW	保留
0 号内存控制器 ECC 计数寄存器 Mc0_ecc_cnt	0x0610	RW	0 号内存控制器 ECC 计数寄存器 [7:0]：MC0 int_cnt，配置 ECC 校验触发中断次数阈值 [15:8]：MC0 int_cnt_err(RO)，ECC 校验一位出错次数统计（只读） [23:16]：MC0 int_cnt_fatal(RO)，ECC 校验两位出错次数统计（只读）
0 号内存控制器 ECC 出错次数统计寄存器 Mc0_ecc_cs_cnt	0x0618	RO	0 号内存控制器 ECC 出错次数统计寄存器 [7:0]：MC0 ecc_cnt_cs_0，CS0 出现 ECC 校验错次数统计 [15:8]：MC0 ecc_cnt_cs_1，CS1 出现 ECC 校验错次数统计 [23:16]：MC0 ecc_cnt_cs_2，CS2 出现 ECC 校验错次数统计 [31:24]：MC0 ecc_cnt_cs_3，CS3 出现 ECC 校验错次数统计 [39:32]：MC0 ecc_cnt_cs_4，CS4 出现 ECC 校验错次数统计 [47:40]：MC0 ecc_cnt_cs_5，CS5 出现 ECC 校验错次数统计 [55:48]：MC0 ecc_cnt_cs_6，CS6 出现 ECC 校验错次数统计 [63:56]：MC0 ecc_cnt_cs_7，CS7 出现 ECC 校验错次数统计
0 号内存控制器 ECC 校验码寄存器 Mc0_ecc_code	0x0620	RO	0 号内存控制器 ECC 校验码寄存器 [7:0]：MC0 ecc_code_64，64 位 ECC 校验时的 ECC 校验码，使能内存目录功能时无效 [41:32]：MC0 ecc_code_256，256 位 ECC 校验时的 ECC 校验码，使能内存目录功能时有效 [52:48]：MC0 ecc_code_dir，内存目录 ECC 校验码，只有使能内存目录功能时有效 [60:56]：MC0 ecc_data_dir，内存目录 ECC 数据，只有使能内存目录功能时有效
0 号内存控制器 ECC 出错地址寄存器 Mc0_ecc_addr	0x0628	RO	0 号内存控制器 ECC 出错地址寄存器 [63:0]：MC0 ecc_addr，ECC 校验出错的地址信息
0 号内存控制器 ECC 出错数据寄存器 0 Mc0_ecc_data0	0x0630	RO	0 号内存控制器 ECC 出错数据寄存器 0 [63:0]:Mc0_ecc_data0，ECC 校验出错时的数据信息，64 位 ECC 模式下的数据，256 位 ECC 模式下的数据[63:0]
0 号内存控制器 ECC 出错数据寄存器 1 Mc0_ecc_data1	0x0638	RO	0 号内存控制器 ECC 出错数据寄存器 1 [63:0]:Mc0_ecc_data1，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[127:64]

0 号内存控制器 ECC 出错数据寄存器 2 Mc0_ecc_data2	0x0640	RO	0 号内存控制器 ECC 出错数据寄存器 2 [63:0]:Mc0_ecc_data2，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[191:128]
0 号内存控制器 ECC 出错数据寄存器 3 Mc0_ecc_data3	0x0648	RO	0 号内存控制器 ECC 出错数据寄存器 3 [63:0]:Mc0_ecc_data3，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[255:192]

表 13-4 1 号内存控制器出错状态观测寄存器

寄存器	地址	控制	说明
1 号内存控制器 ECC 设置寄存器 Mc1_ecc_set	0x0700	RW	1 号内存控制器 ECC 设置寄存器 [5:0]：MC1 int_enable，中断使能 [8]：MC1 int_trigger，中断触发配置 [21:16]：MC1 int_vector(RO)，中断向量（只读） [33:32]：MC1 ecc_enable，ECC 相关功能使能 [40]：MC1 rd_before_wr，读后写功能使能
	0x0708	RW	保留
1 号内存控制器 ECC 计数寄存器 Mc1_ecc_cnt	0x0710	RW	1 号内存控制器 ECC 计数寄存器 [7:0]：MC1 int_cnt，配置 ECC 校验触发中断次数阈值 [15:8]：MC1 int_cnt_err(RO)，ECC 校验一位出错次数统计（只读） [23:16]：MC1 int_cnt_fatal(RO)，ECC 校验两位出错次数统计（只读）
1 号内存控制器 ECC 出错次数统计寄存器 Mc1_ecc_cs_cnt	0x0718	RO	1 号内存控制器 ECC 出错次数统计寄存器 [7:0]：MC1 ecc_cnt_cs_0，CS0 出现 ECC 校验错次数统计 [15:8]：MC1 ecc_cnt_cs_1，CS1 出现 ECC 校验错次数统计 [23:16]：MC1 ecc_cnt_cs_2，CS2 出现 ECC 校验错次数统计 [31:24]：MC1 ecc_cnt_cs_3，CS3 出现 ECC 校验错次数统计 [39:32]：MC1 ecc_cnt_cs_4，CS4 出现 ECC 校验错次数统计 [47:40]：MC1 ecc_cnt_cs_5，CS5 出现 ECC 校验错次数统计 [55:48]：MC1 ecc_cnt_cs_6，CS6 出现 ECC 校验错次数统计 [63:56]：MC1 ecc_cnt_cs_7，CS7 出现 ECC 校验错次数统计
1 号内存控制器 ECC 校验码寄存器 Mc1_ecc_code	0x0720	RO	1 号内存控制器 ECC 校验码寄存器 [7:0]：MC1 ecc_code_64，64 位 ECC 校验时的 ECC 校验码，使能内存目录功能时无效 [41:32]：MC1 ecc_code_256，256 位 ECC 校验时的 ECC 校验码，使能内存目录功能时有效 [52:48]：MC1 ecc_code_dir，内存目录 ECC 校验码，只有使能内存目录功能时有效 [60:56]：MC1 ecc_data_dir，内存目录 ECC 数据，只有使能内存目录功能时有效
1 号内存控制器 ECC 出错地址寄存器 Mc1_ecc_addr	0x0728	RO	1 号内存控制器 ECC 出错地址寄存器 [63:0]：MC1 ecc_addr，ECC 校验出错的地址信息
1 号内存控制器 ECC 出错数据寄存器 0 Mc1_ecc_data0	0x0730	RO	1 号内存控制器 ECC 出错数据寄存器 0 [63:0]:Mc1_ecc_data0，ECC 校验出错时的数据信息，64 位 ECC 模式下的数据，256 位 ECC 模式下的数据[63:0]
1 号内存控制器 ECC 出错数据寄存器 1 Mc1_ecc_data1	0x0738	RO	1 号内存控制器 ECC 出错数据寄存器 1 [63:0]:Mc1_ecc_data1，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[127:64]

1 号内存控制器 ECC 出错数据寄存器 2 Mc1_ecc_data2	0x0740	RO	1 号内存控制器 ECC 出错数据寄存器 2 [63:0]:Mc1_ecc_data2，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[191:128]
1 号内存控制器 ECC 出错数据寄存器 3 Mc1_ecc_data3	0x0748	RO	1 号内存控制器 ECC 出错数据寄存器 3 [63:0]:Mc1_ecc_data3，ECC 校验出错时的数据信息，256 位 ECC 模式下的数据[255:192]

14 HyperTransport 控制器

龙芯 3A4000 中，HyperTransport 总线用于实现外部设备连接以及多芯片互连。用于外设连接时，可由用户程序自由选择是否支持 IO Cache 一致性（通过地址窗口 Uncache 进行设置，详见 14.5.14 节）：当配置为支持 Cache 一致性模式时，IO 设备对内 DMA 的访问对于 Cache 层次透明，即由硬件自动维护其一致性,而无需软件通过程序 Cache 指令进行维护；当 HyperTransport 总线用于多芯片互连时，HT0 控制器(初始地址为 0x0C00_0000_0000 – 0x0DFF_FFFF_FFFF)可通过引脚配置来支持片间 Cache 一致性传输，而 HT1 控制器(初始地址为 0x0E00_0000_0000 – 0x0FFF_FFFF_FFFF)可通过软件配置来支持片间 Cache 一致性维护，详见 14.7 节。在 8 片互连结构中，HT1_HI 控制器的一致性模式通过 CHIP_CONFIG 中的引脚进行配置。

HyperTransport 控制器最高支持双向 16 位宽度以及 2.4GHz 运行频率。在系统自动初始化建立连接后，用户程序可以通过修改协议中相应的配置寄存器，实现对宽度和运行频率的更改，并重新进行初始化，具体方法见 14.1 节。

龙芯 3A4000 HyperTransport 控制器的主要特征如下：

- 支持 HT1.0/HT3.0 协议
- 支持 200/400/800/1600/2000/2400MHz 运行频率
- 控制器频率最高为 1GHz
- HT1.0 支持 8 位宽度
- HT3.0 支持 8/16 位宽度
- 每个 HT 控制器（HT0/HT1）可以配置为两个 8 位 HT 控制器
- 总线控制信号（包括 PowerOK，Rstn，LDT_Stopn）方向可配置
- 外设 DMA 空间 Cache/Uncache 可配置
- 用于多片互连时可配置为 Cache 一致性模式

14.1 HyperTransport 硬件设置及初始化

HyperTransport 总线由传输信号总线和控制信号引脚等组成，下表给出了 HyperTransport 总线相关的引脚及其功能描述。

表 14-1 HyperTransport 总线相关引脚信号

引脚	名称	描述
HT0_8x2	总线宽度配置	1：将 16 位 HyperTransport 总线配置为两个独立的 8 位总线，分别由两个独立的控制器控制，地址空间的区分为 HT0_Lo：address[40] = 0； HT0_Hi：address[40] = 1; 0：将 16 位 HyperTransport 总线作为一个 16 位总线使用，由 HT0_Lo 控制，地址空间为 HT0_Lo 的地址，即 address[40] =0；HT0_Hi 所有信号无效。
HT0_Lo_mode	主设备模式	1：将 HT0_Lo 设为主设备模式，这个模式下，总线控制信号等由 HT0_Lo 驱动，这些控制信号包括 HT0_Lo_Powerok，HT0_Lo_Rstn，HT0_Lo_Ldt_Stopn。这个模式下，这些控制信号也可以为双向驱动。同时这个引脚决定（取反）寄存器“Act as Slave”的初始值，这个寄存器为 0 时，HyperTransport 总线上的包中的 Bridge 位为 1，否则为 0。另外，这个寄存器为 0 时，如果 HyperTransport 总线上的请求地址没有在控制器的接收窗口命中时，将作为 P2P 请求重新发回总线，如果这个寄存器为 1 时，没有命中，则作为错误请求做出响应。 0：将 HT0_Lo 设为从设备模式，这个模式下，总线控制信号等由对方设备驱动，这些控制信号包括 HT0_Lo_Powerok，HT0_Lo_Rstn，HT0_Lo_Ldt_Stopn。这个模式下，这些控制信号由对方设备驱动，如果没有被正确驱动，则 HT 总线不能正确工作。
HT0_Lo_Powerok	总线 Powerok	HyperTransport 总线 Powerok 信号， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。
HT0_Lo_Rstn	总线 Rstn	HyperTransport 总线 Rstn 信号， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。
HT0_Lo_Ldt_Stopn	总线 Ldt_Stopn	HyperTransport 总线 Ldt_Stopn 信号， HT0_Lo_Mode 为 1 时，由 HT0_Lo 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。
HT0_Lo_Ldt_Reqn	总线 Ldt_Reqn	HyperTransport 总线 Ldt_Reqn 信号，
HT0_Hi_mode	主设备模式	1：将 HT0_Hi 设为主设备模式，这个模式下，总线控制信号等由 HT0_Hi 驱动，这些控制信号包括 HT0_Hi_Powerok，HT0_Hi_Rstn，HT0_Hi_Ldt_Stopn。这个模式下，这些控制信号也可以为双向驱动。同时这个引脚决定（取反）寄存器“Act as Slave”的初始值，这个寄存器为 0 时，HyperTransport 总线上的包中的 Bridge 位为 1，否则为 0。另外，这个寄存器为 0 时，如果 HyperTransport 总线上的请求地址没有在控制器的接收窗口命中时，将作为 P2P 请求重新发回总线，如果这个寄存器为 1 时，没有命中，则作为错误请求做出响应。 0：将 HT0_Hi 设为从设备模式，这个模式下，总线控制信号等由对方设备驱动，这些控制信号包括 HT0_Hi_Powerok，HT0_Hi_Rstn，HT0_Hi_Ldt_Stopn。这个模式下，这些控制信

		号由对方设备驱动，如果没有被正确驱动，则 HT 总线不能正确工作。
HT0_Hi_Powerok	总线 Powerok	HyperTransport 总线 Powerok 信号， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Hi_Rstn	总线 Rstn	HyperTransport 总线 Rstn 信号， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Hi_Ldt_Stopn	总线 Ldt_Stopn	HyperTransport 总线 Ldt_Stopn 信号， HT0_Lo_Mode 为 1 时，由 HT0_Hi 控制； HT0_Lo_Mode 为 0 时，由对方设备控制。 HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Hi_Ldt_Reqn	总线 Ldt_Reqn	HyperTransport 总线 Ldt_Reqn 信号， HT0_8x2 为 1 时，控制高 8 位总线； HT0_8x2 为 0 时，无效。
HT0_Rx_CLKp[1:0] HT0_Rx_CLKn[1:0] HT0_Tx_CLKp[1:0] HT0_Tx_CLKp[1:0]	CLK[1:0]	HyperTransport 总线 CLK 信号 HT0_8x2 为 1 时，CLK[1]由 HT0_Hi 控制 CLK[0]由 HT0_Lo 控制 HT0_8x2 为 0 时，CLK[1:0]由 HT0_Lo 控制
HT0_Rx_CTLp[1:0] HT0_Rx_CTLn[1:0] HT0_Tx_CTLp[1:0] HT0_Tx_CTLn[1:0]	CTL[1:0]	HyperTransport 总线 CTL 信号 HT0_8x2 为 1 时，CTL[1]由 HT0_Hi 控制 CTL[0]由 HT0_Lo 控制 HT0_8x2 为 0 时，CTL[1]无效 CTL[0]由 HT0_Lo 控制
HT0_Rx_CADp[15:0] HT0_Rx_CADn[15:0] HT0_Tx_CADp[15:0] HT0_Tx_CADn[15:0]	CAD[15:0]	HyperTransport 总线 CAD 信号 HT0_8x2 为 1 时，CAD[15:8]由 HT0_Hi 控制 CAD[7:0]由 HT0_Lo 控制 HT0_8x2 为 0 时，CAD[15:0]由 HT0_Lo 控制

HyperTransport 的初始化在每次复位完成后自动开始，冷启动后 HyperTransport 总线将自动工作在最低频率 (200MHz)与最小宽度(8bit)，并尝试进行总线初始化握手。初始化是否已处于完成状态可以由寄存器“Init Complete”（见 14.5.2 节）读出。初始化完成后，总线的宽度可以由寄存器“Link Width Out”与“Link Width In” （见 14.5.2 节）读出。初始化完成后，用户可重写寄存器“Link Width Out”、“Link Width In”以及“Link Freq”，同时还需要配置对方设备的相应寄存器，配置完成后需要热复位总线或者通过“HT_Ldt_Stopn”信号进行重新初始化操作，以便使寄存器重写后的值生效。重新初始化完成后 HyperTransport 总线将工作在新的频率和宽度。需要注意的是，

HyperTransport 两端的设备的配置需要一一对应，否则将使得 HyperTransport 接口不能正常工作。

14.2 HyperTransport 协议支持

龙芯 3A4000 的 HyperTransport 总线支持 1.03/3.0 版协议中的大部分命令，并且在支持多芯片互连的扩展一致性协议中加入了一些扩展指令。在以上两种模式下，HyperTransport 接收端可接收的命令如下表所示。需要注意的是，不支持 HyperTransport 总线的原子操作命令。

表 14-2 HyperTransport 接收端可接收的命令

编码	通道	命令	标准模式	扩展（一致性）
000000	-	NOP	空包或流控	
000001	NPC	FLUSH	无操作	
x01xxx	NPC or PC	Write	bit 5：0 - Nonposted 1 - Posted bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：Don't Care	bit 5：必为 1，POSTED bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：必为 1
01xxxx	NPC	Read	bit 3：Don't Care bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：Don't Care	bit 3：Don't Care bit 2：0 – Byte 1 – Doubleword bit 1：Don't Care bit 0：必为 1
110000	R	RdResponse	读操作返回	
110011	R	TgtDone	写操作返回	
110100	PC	WrCoherent	----	写命令扩展
110101	PC	WrAddr	----	写地址扩展
111000	R	RespCoherent	----	读响应扩展
111001	NPC	RdCoherent	----	读命令扩展
111010	PC	Broadcast	无操作	
111011	NPC	RdAddr	----	读地址扩展
111100	PC	FENCE	保证序关系	
111111	-	Sync/Error	Sync/Error	

对于发送端，在两种模式下会向外发送的命令如下表所示。

表 14-3 两种模式下会向外发送的命令

编码	通道	命令	标准模式	扩展（一致性）
000000	-	NOP	空包或流控	
x01x0x	NPC or PC	Write	bit 5：0 - Nonposted 1 - Posted bit 2：0 – Byte 1 – Doubleword bit 0：必为 0	bit 5：必为 1，POSTED bit 2：0 – Byte 1 – Doubleword bit 0：必为 1
010x0x	NPC	Read	bit 2：0 – Byte 1 – Doubleword bit 0：Don't Care	bit 2：0 – Byte 1 – Doubleword bit 0：必为 1
110000	R	RdResponse	读操作返回	

110011	R	TgtDone	写操作返回	
110100	PC	WrCoherent	----	写命令扩展
110101	PC	WrAddr	----	写地址扩展
111000	R	RespCoherent	----	读响应扩展
111001	NPC	RdCoherent	----	读命令扩展
111011	NPC	RdAddr	----	读地址扩展
111111	-	Sync/Error	只会转发	

14.3 HyperTransport 中断支持

HyperTransport 控制器提供了 256 个中断向量，可以支持 Fix，Arbiter 等类型的中断，但是，没有对硬件自动 EOI 提供支持。对于以上两种支持类型的中断，控制器在接收之后会自动写入中断寄存器中，并根据中断屏蔽寄存器的设置对系统中断控制器进行中断通知。具体的中断控制请见 14.5.7 节中的中断控制寄存器描述。

14.3.1 PIC 中断

控制器对 PIC 中断做了专门的支持，以加速该类型的中断处理。

一个典型的 PIC 中断由下述步骤完成：①PIC 控制器向系统发送 PIC 中断请求；②系统向 PIC 控制器发送中断向量查询；③PIC 控制器向系统发送中断向量号；④系统清除 PIC 控制器上的对应中断。只有上述 4 步都完成后，PIC 控制器才会对系统发出下一个中断。对于龙芯 3A4000 HyperTransport 控制器，将自动进行前 3 步的处理，并将 PIC 中断向量写入 256 个中断向量中的对应位置。而软件系统在处理了该中断之后，需要进行第 4 步处理，即向 PIC 控制器发出清中断。之后开始下一个中断的处理过程。

14.3.2 本地中断处理

传统的中断处理模式中，所有的中断都由 HT 控制器内部的中断向量进行存储，再通过 HT 控制器的中断线连接至芯片上的中断路由器进行分发。这种情况下，HT 中断只通过有限的几种连接方式对 CPU 核进行中断，也不能进行跨片分发，使用场景比较受限。

在这种 HT 中断模式下，进行中断处理时，芯片上的中断路由器对软件透明，内核直接到 HT 控制器的中断向量（一般为 0x90000efdfb000080）上进行查找，然后按位进行处理，此时无论路由模式如何配置，都是直接读到 HT 控制器上的所有中断。

14.3.3 扩展中断处理

在 3A4000 中实现的扩展 IO 中断，可以大大增加中断分发、中断处理的灵活性。

HT 的中断扩展模式下，将除了 PIC 中断之外的其他中断直接写入芯片中断路由器上新增的扩展中断寄存器上，再根据扩展中断寄存器的相关配置进行路由或者分发。

使用扩展 IO 中断之后，进行中断处理时，HT 控制器对软件透明，内核直接到扩展 IO 状态寄存器（配置空间 0x1800）上读取中断状态进行处理，每个核只会读到中断自己的中断状态并进行处理，不同核之间不会产生干扰。

HT 控制器上是通过使能外部中断转换配置寄存器来进行中断转发的。如 14.5.34 所述，软件需要将 HT_int_trans 设置为扩展 IO 中断触发寄存器的目标地址。3A4000 中该寄存器地址为 0x1fe01140，或 0x10000_00001140。

内核在使用扩展中断处理前，需要使能“其他功能设置寄存器”中的对应位。该寄存器基地址为 0x1fe00000，偏移地址 0x0420。

表 14-4 其它功能设置寄存器

位域	字段名	访问	复位值	描述
51:48	EXT_INT_en	RW	0x0	扩展 IO 中断使能

14.4 HyperTransport 地址窗口

14.4.1 HyperTransport 空间

龙芯 3A4000 处理器中，默认的 4 个 HyperTransport 接口的地址窗口分布如下：

表 14-5 默认的 4 个 HyperTransport 接口的地址窗口分布

基地址	结束地址	大小	定义
0x0A00_0000_0000	0x0AFF_FFFF_FFFF	1 Tbytes	HT0_LO 窗口
0x0B00_0000_0000	0x0BFF_FFFF_FFFF	1 Tbytes	HT0_HI 窗口
0x0E00_0000_0000	0x0EFF_FFFF_FFFF	1 Tbytes	HT1_LO 窗口
0x0F00_0000_0000	0x0FFF_FFFF_FFFF	1 Tbytes	HT1_HI 窗口

在默认情况下（未对系统地址窗口另行配置），软件根据上述地址空间对各个 HyperTransport 接口进行访问，此外，软件还可以通过对交叉开关上的地址窗口进行配置实现用其它的地址空间对其进行访问（详见 3.3 节）。每个 HyperTransport 接口的内部 40 位地址空间其地址窗口分布如下表所示。

表 14-6 龙芯 3 号处理器 HyperTransport 接口内部的地址窗口分布

基地址	结束地址	大小	定义
0x00_0000_0000	0xFC_FFFF_FFFF	1012 Gbytes	MEM 空间
0xFD_0000_0000	0xFD_F7FF_FFFF	3968 Mbytes	保留
0xFD_F800_0000	0xFD_F8FF_FFFF	16 Mbytes	中断

0xFD_F900_0000	0xFD_F90F_FFFF	1 Mbyte	PIC 中断响应
0xFD_F910_0000	0xFD_F91F_FFFF	1 Mbyte	系统信息
0xFD_F920_0000	0xFD_FAFF_FFFF	30 Mbytes	保留
0xFD_FB00_0000	0xFD_FBFF_FFFF	16 Mbytes	HT 控制器配置空间
0xFD_FC00_0000	0xFD_FDFF_FFFF	32 Mbytes	I/O 空间
0xFD_FE00_0000	0xFD_FFFF_FFFF	32 Mbytes	HT 总线配置空间
0xFE_0000_0000	0xFF_FFFF_FFFF	8 Gbytes	保留

14.4.2 HyperTransport 控制器内部窗口配置

龙芯3A4000处理器的HyperTransport接口中提供了多种丰富的地址窗口供用户使用，这些地址窗口的作用和功能描述如下表所示。

表 14-7 龙芯 3A4000 处理器 HyperTransport 接口中提供的地址窗口

地址窗口	窗口数	接受总线	作用	备注
接收窗口（窗口配置见 14.5.10 节）	3	HyperTransport	判断是否接收 HyperTransport 总线上发出的访问。	处于主桥模式时（即配置寄存器中 act_as_slave 为 0），只有落在这些地址窗口中的访问会被内部总线所响应，其它访问将会被认为是 P2P 访问重新发回到 HyperTransport 总线上；处于设备模式时（即配置寄存器中 act_as_slave 为 1），只有落在这些地址窗口中的访问会被内部总线所接收并处理，其它访问将会按照协议给出错误返回。
Post 窗口（窗口配置见 14.5.12 节）	2	内部总线	判断是否将内部总线对 HyperTransport 总线的写访问作为 Post Write	落在这些地址空间中的对外写访问将作为 Post Write。 Post Write：HyperTransport 协议中，这种写访问不需要等待写完成响应，即在控制器向总线发出这个写访问之后就将对处理器进行写访问完成响应。
可预取窗口（窗口配置见 14.5.13 节）	2	内部总线	判断是否接收内部的 Cache 访问，取指访问。	当处理器核乱序执行时，会对总线发出一些猜测读访问或是取指访问，这种访问对于某些 IO 空间是错误的。在默认情况下，这种访问 HT 控制器将直接返回而不对 HyperTransport 总线进行访问。通过这些窗口可以使能对 HyperTransport 总线的这类访问。
Uncache 窗口（窗口配置见 14.5.14 节）	2	HyperTransport	判断是否将 HyperTransport 总线上的访问作为对内部的 Uncache 访问	龙芯 3A4000 处理器内部的 IO DMA 访问，在默认情况下将作为 Cache 方式访问经由 SCache 判断是否命中，从而维护其 IO 一致性信息。而通过这些窗口的配置，可以使在这些窗口命中的访问以 Uncache 的方式直接访问内存，而不通过硬件维护其 IO 一致性信息。

14.5 配置寄存器

配置寄存器模块主要用于控制从 AXI SLAVE 端或是 HT RECEIVER 端到达的配置寄存器访问请求，进行外部中断处理，并保存了大量软件可见的用于控制系统各种工作方式的配置寄存器。

首先，用于控制 HT 控制器各种行为的配置寄存器的访问与存储都在本模块中，本模块的访问偏移地址在 HT 控制器端为 0xFD_FB00_0000 到 0xFD_FBFF_FFFF。HT 控制器中所有软件可见寄存器如下表所示：

Enable	0x00	Device ID		Vendor ID	
	0x04	Status		Command	
	0x08	Class Code			Revision ID
	0x0c	BIST	Header Type	Latency Timer	Cache Line Size
	0x10				
	0x14				
	0x18				
	0x1c				
	0x20				
	0x24				
	0x28	Cardbus CIS Pointer			
	0x2c	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Enable Address			
	0x34	Reserved			Capabilities Pointer
	0x38	Reserved			
	0x3c	Bridge Control		Interrupt Pin	Interrupt Line
Cap 0 PRI	0x40	Command		Capabilities Pointer	Capability ID
	0x44	Link Config 0		Link Control 0	
	0x48	Link Config 1		Link Control 1	
	0x4C	LinkFreqCap0		Link Error0/Link Freq 0	Revision ID
	0x50	LinkFreqCap1		Link Error1/Link Freq 1	Feature
	0x54	Error Handling		Enumeration Scratchpad	
	0x58	Reserved		Mem Limit Upper	Mem Enable Upper
Cap 1 Retry	0x60	Capability Type	Reserved	Capability Pointer	Capabiliter ID
	0x64	Status 1	Control 1	Status 0	Control 0
	0x68	Retry Count 1		Retry Count 0	

CAP 3	0x6C	Capability Type	Revision ID	Capability Pointer	Capabiliter ID
CAP 4 Interrupt	0x70	Capability Type	Index	Capability Pointer	Capabiliter ID
	0x74	Dataport			
	0x78	IntrInfo[31:0]			
	0x7C	IntrInfo[63:32]			
Int Vector	0x80	INT Vector[31:0]			
	0x84	INT Vector[63:32]			
	0x88	INT Vector[95:64]			
	0x8C	INT Vector[127:96]			
	0x90	INT Vector[159:128]			
	0x94	INT Vector[191:160]			
	0x98	INT Vector[223:192]			
	0x9C	INT Vector[255:224]			
	0xA0	INT Enable[31:0]			
	0xA4	INT Enable[63:32]			
	0xA8	INT Enable[95:64]			
	0xAC	INT Enable[127:96]			
	0xB0	INT Enable[159:128]			
	0xB4	INT Enable[191:160]			
	0xB8	INT Enable[223:192]			
	0xBC	INT Enable[255:224]			
CAP 5 Gen3	0xC0	Capability Type	Cap Enum/Index	Capability Pointer	Capabiliter ID
	0xC4	Global Link Training			
	0xC8	Transmitter Configuration 0			
	0xCC	Receiver Configuration 0			
	0xD0	Link Training 0			
	0xD4	Frequency Extension			
	0xD8	Transmitter Configuration 1			
	0xDC	Receiver Configuration 1			
	0xE0	Link Training 1			
	0xE4	BIST Control			

Enable	0x100	Device ID		Vendor ID	
	0x104	Status		Command	
	0x108	Class Code			Revision ID
	0x10c	BIST	Header Type	Latency Timer	Cache Line Size
	0x110				
	0x114				
	0x118				
	0x11c				

	0x120			
	0x124			
	0x128	Cardbus CIS Pointer		
	0x12c	Subsystem ID	Subsystem Vendor ID	
	0x130	Expansion ROM Enable Address		
	0x134	Reserved		Capabilities Pointer
	0x138	Reserved		
	0x13c	Bridge Control	Interrupt Pin	Interrupt Line
Receive Windows	0x140	HT RX Enable 0		
	0x144	HT RX Mask 0		
	0x148	HT RX Enable 1		
	0x14C	HT RX Mask 1		
	0x150	HT RX Enable 2		
	0x154	HT RX Mask 2		
	0x158	HT RX Enable 3		
	0x15C	HT RX Mask 3		
	0x160	HT RX Enable 4		
	0x164	HT RX Mask 4		
Header Trans	0x168	HT RX Header Trans		
	0x16C	HT RX EXT Header Trans		
Post Windows	0x170	HT TX Post Enable 0		
	0x174	HT TX Post Mask 0		
	0x178	HT TX Post Enable 1		
	0x17C	HT TX Post Mask 1		
Prefetchable Windows	0x180	HT TX Prefetchable Enable 0		
	0x184	HT TX Prefetchable Mask 0		
	0x188	HT TX Prefetchable Enable 1		
	0x18C	HT TX Prefetchable Mask 1		
Uncache Windows	0x190	HT RX Uncache Enable 0		
	0x194	HT RX Uncache Mask 0		
	0x198	HT RX Uncache Enable 1		
	0x19C	HT RX Uncache Mask 1		
	0x1A0	HT RX Uncache Enable 2		
	0x1A4	HT RX Uncache Mask 2		
	0x1A8	HT RX Uncache Enable 3		
	0x1AC	HT RX Uncache Mask 3		
P2P Windows	0x1B0	HT RX P2P Enable 0		
	0x1B4	HT RX P2P Mask 0		
	0x1B8	HT RX P2P Enable 1		
	0x1BC	HT RX P2P Mask 1		

APP Config	0x1C0	APP Configuration 0
	0x1C4	APP Configuration 1
	0x1C8	RX Bus Value
	0x1CC	PHY status
Buffer	0x1D0	TX Buffer 0
	0x1D4	TX Buffer 1 / Rx buffer hi
	0x1D8	TX Buffer turning
	0x1DC	RX Buffer lo
Training	0x1E0	Training 0 Counter Short
	0x1E4	Training 0 Counter Long
	0x1E8	Training 1 Counter
	0x1EC	Training 2 Counter
	0x1F0	Training 3 Counter
PLL	0x1F4	PLL Configuration
PHY	0x1F8	IO Configuration
	0x1FC	PHY Configuration

DEBUG	0x240	HT3 DEBUG 0
	0x244	HT3 DEBUG 1
	0x248	HT3 DEBUG 2
	0x24C	HT3 DEBUG 3
	0x250	HT3 DEBUG 4
	0x254	HT3 DEBUG 5
	0x258	HT3 DEBUG 6
POST ID WINDOWS	0x260	HT TX POST ID WIN0
	0x264	HT TX POST ID WIN1
	0x268	HT TX POST ID WIN2
	0x26C	HT TX POST ID WIN3
POST ID WINDOWS	0x270	INT TRANS WIN lo
	0x274	INT TRANS WIN hi

每个寄存器的具体含义如下节所示：

14.5.1 Bridge Control

偏移量：　　0x3C
复位值：　　0x00000000
名称：　　Bus Reset Control

表 14-8 Bus Reset Control 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:23	Reserved	9	0x0		保留
22	Reset	1	0x0	R/W	总线复位控制： 0->1：HT_RSTn 置 0，总线复位 1->0：HT_RSTn 置 1，总线解复位
21:0	Reserved	22	0x0		保留

14.5.2 Capability Registers

偏移量：　0x40
复位值：　0x20010008
名称：　Command，Capabilities Pointer，Capability ID

表 14-9 Command，Capabilities Pointer，Capability ID 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:29	Slave/Pri	3	0x0	R	Command 格式为 HOST/Sec
28:26	Reserved	2	0x0	R	保留
25:21	Unit Count	5	0x0	R/W	提供给软件用于记录当前的 Unit 个数
20:16	Unit ID	5	0x0		HOST 模式时：可用于记录使用 ID 个数 SLAVE 模式时：记录自身 Unit ID HOST/SLAVE 模式由 act_as_slave 寄存器控制
15:08	Capabilities Pointer	8	0x60	R	下一个 Cap 寄存器偏移地址
7:0	Capability ID	8	0x08	R	HyperTransport capability ID

偏移量：　0x44
复位值：　0x00112000
名称：　Link Config，Link Control

表 14-10 Link Config，Link Control 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
30:28	Link Width Out	3	0x0	R/W	发送端宽度 冷复位后的值为当前连接的最大宽度，写入此寄存器的值将会在下次热复位或是 HT Disconnect 之后生效 000：8 位方式 001：16 位方式
27	Reserved	1	0x0		保留
26:24	Link Width In	3	0x0	R/W	接收端宽度

					冷复位后的值为当前连接的最大宽度，写入此寄存器的值将会在下次热复位或是 HT Disconnect 之后生效
23	Dw Fc out	1	0x0	R	发送端不支持双字流控
22:20	Max Link Width out	3	0x1	R	HT 总线发送端最大宽度：16bits
19	Dw Fc In	1	0x0	R	接收端不支持双字流控
18:16	Max Link Width In	3	0x1	R	HT 总线接收端最大宽度：16bits
15:14	Reserved	2	0x0		保留
13	LDTSTOP# Tristate Enable	1	0x1	R/W	当 HT 总线进入 HT Disconnect 状态时，是否关闭 HT PHY 1：关闭 0：不关闭
12:10	Reserved	3	0x0		保留
9	CRC Error (hi)	1	0x0	R/W	高 8 位发生 CRC 错
8	CRC Error (lo)	1	0x0	R/W	低 8 位发生 CRC 错
7	Trans off	1	0x0	R/W	HT PHY 关闭控制 处于 16 位总线工作方式时 1：关闭 高/低 8 位 HT PHY 0：使能 低 8 位 HT PHY，高 8 位 HT PHY 由 bit 0 控制
6	End of Chain	0	0x0	R	HT 总线末端
5	Init Complete	1	0x0	R	HT 总线初始化是否完成
4	Link Fail	1	0x0	R	指示连接失败
3:2	Reserved	2	0x0		保留
1	CRC Flood Enable	1	0x0	R/W	发生 CRC 错误时，是否 flood HT 总线
0	Trans off (hi)	1	0x0	R/W	使用 16 位 HT 总线运行 8 位协议时，高 8 位 PHY 关闭控制 1：关闭 高 8 位 HT PHY 0：使能 高 8 位 HT PHY

偏移量： 0x4C
复位值： 0x80250023
名称： Revision ID，Link Freq，Link Error，Link Freq Cap

表 14-11 Revision ID，Link Freq，Link Error，Link Freq Cap 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	Link Freq Cap	16	0x0000	R	支持的 HT 总线频率，根据外部 PLL 的设置产生不同的值（当使用软件配置 PLL（0x1F4）时，该位无意义） {3.2G,2.6G,2.4G,2.2G,2.0G,1.8G,1.6G,

					1.4G,1.2G,1.0G,800M,600M,500M,400M,300M,200M}
15:14	Reserved	2	0x0		保留
13	Over Flow Error	1	0x0	R	HT 总线包溢出
12	Protocol Error	1	0x0	R/W	协议错误，指 HT 总线上收到不可识别的命令
11:8	Link Freq	4	0x0	R/W	HT 总线工作频率，写入此寄存器的值后将在下次热复位或是 HT Disconnect 之后生效，设置的值与 Link Freq Cap 的位相对应 （当使用软件配置 PLL（0x1F4）时，该位无意义）
7:0	Revision ID	8	0x60	R/W	版本号：3.0

偏移量： 0x50
复位值： 0x00000002
名称： Feature Capability

表 14-12 Feature Capability 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:9	Reserved	23	0x0		保留
8	Extended Register	1	0x0	R	没有
7:4	Reserved	3	0x0		保留
3	Extended CTL Time	1	0x0	R	不需要
2	CRC Test Mode	1	0x0	R	不支持
1	LDTSTOP#	1	0x1	R	支持 LDTSTOP#
0	Isochronous Mode	1	0x0	R	不支持

14.5.3 Error Retry 控制寄存器

用于 HyerTransport 3.0 模式下错误重传使能，配置 Short Retry 的最大次数，显示 Retry 计数器是否翻转。

偏移量： 0x64
复位值： 0x00000000
名称： Error Retry 控制寄存器

表 14-13 Error Retry 控制寄存器

位域	位域名称	位宽	复位值	访问	描述
31:10	Reserved	22	0x0	R	保留

9	Retry Count Rollover	1	0x0	R	Retry 计数器计数翻转
8	Reserved	1	0x0	R	保留
7:6	Short Retry Attempts	2	0x0	R/W	允许的最大 Short Retry 次数
5:1	Reserved	5	0x0	R	
0	Link Retry Enable	1	0x0	R/W	出错重连功能使能控制

14.5.4 Retry Count 寄存器

用于 HyerTransport 3.0 模式下错误重传计数。

偏移量： 0x68
复位值： 0x00000000
名称： Retry Count 寄存器

表 14-14 Retry Count 寄存器

位域	位域名称	位宽	复位值	访问	描述
31:16	Reserved	16	0x0	R	保留
15:0	Retry Count	16	0x0	R	Retry 计数

14.5.5 Revision ID 寄存器

用于配置控制器版本,配置成新的版本号，通过 Warm Reset 生效。

偏移量： 0x6C
复位值： 0x00200000
名称： RevisionID 寄存器

表 14-15 Revision ID 寄存器

位域	位域名称	位宽	复位值	访问	描述
31:24	Reserved	8	0x0	R	保留
23:16	Revision ID	8	0x20	R/W	Revision ID 控制寄存器 0x20：HyperTransport 1.00 0x60：HyperTransport 3.00
15:0	Reserved	16	0x0	R	保留

14.5.6 Interrupt Discovery & Configuration

偏移量： 0x70
复位值： 0x80000008
名称： Interrupt Capability

表 14-16 Interrupt Capability 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:24	Capabilities Pointer	8	0x80	R	Interrupt discovery and configuration block
23:16	Index	8	0x0	R/W	读寄存器偏移地址
15:8	Capabilities Pointer	8	0x0	R	Capabilities Pointer
7:0	Capability ID	8	0x08	R	Hypertransport Capablity ID

偏移量： 0x74
复位值： 0x00000000
名称： Dataport

表 14-17 Dataport 寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:0	Dataport	32	0x0	R/W	当上一寄存器 Index 为 0x10 时，本寄存器读写结果为 0xa8 寄存器，否则为 0xac

偏移量： 0x78
复位值： 0xF8000000
名称： IntrInfo[31:0]

表 14-18 IntrInfo 寄存器定义（1）

位域	位域名称	位宽	复位值	访问	描述
31:24	IntrInfo[31:24]	8	0xF8	R	保留
23:2	IntrInfo[23:2]	22	0x0	R/W	IntrInfo[23:2]，当发出 PIC 中断时，IntrInfo 的值用来表示中断向量
1:0	Reserved	2	0x0	R	保留

偏移量： 0x7c
复位值： 0x00000000
名称： IntrInfo[63:32]

表 14-19 IntrInfo 寄存器定义（2）

位域	位域名称	位宽	复位值	访问	描述
31:0	IntrInfo[63:32]	32	0x0	R	保留

14.5.7 中断向量寄存器

中断向量寄存器共 256 个，其中除去 HT 总线上的 Fix、Arbiter 以及 PIC 中断直接映

射到此 256 个中断向量之中，其它的中断，如 SMI，NMI，INIT，INTA，INTB，INTC，INTD 可以通过寄存器 0x50 的[28:24]映射到任意一个 8 位中断向量上去，映射的顺序为{INTD，INTC，INTB，INTA，1’b0，INIT，NMI，SMI}。此时中断向量对应值为{Interrupt Index，内部向量[2:0]}。

默认情况下可以将 256 位中断分发到 4 位中断线上。在不使用高 8 位 HT 控制器的中断时，也可以通过设置 ht_int_8bit 将 256 位中断分发到 8 位中断线上。

256 个中断向量根据中断路由方式选择不同的寄存器配置映射到不同的中断线上，具体的映射方式为：

中断数	Strip	0	1	2	3	4	5	6	7
4 X = [63:0]	1	[X]	[X+64]	[X+128]	[X+192]	-	-	-	-
	2	[2X]	[2X+1]	[2X+128]	[2X+129]	-	-	-	-
	4	[4X]	[4X+1]	[4X+2]	[4X+3]	-	-	-	-
8 X = [31:0] Y = [63:32]	1	[X]	[Y]	[X+64]	[Y+64]	[X+128]	[Y+128]	[X+192]	[Y+192]
	2	[2X]	[2Y]	[2X+1]	[2Y+1]	[2X+128]	[2Y+128]	[2X+129]	[2Y+129]
	4	[4X]	[4X+32]	[4X+1]	[4X+33]	[4X+2]	[4X+34]	[4X+3]	[4X+35]

以使用 4 位中断线为例，不同的映射方式如下。

ht_int_stripe_1:

[0,1,2,3……63]对应中断线 0 /HT HI 对应中断线 4

[64,65,66,67……127]对应中断线 1 /HT HI 对应中断线 5

[128,129,130,131……191]对应中断线 2 /HT HI 对应中断线 6

[192,193,194,195……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_2:

[0,2,4,6……126]对应中断线 0 /HT HI 对应中断线 4

[1,3,5,7……127]对应中断线 1 /HT HI 对应中断线 5

[128,130,132,134……254]对应中断线 2 /HT HI 对应中断线 6

[129,131,133,135……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_4:

[0,4,8,12……252]对应中断线 0 /HT HI 对应中断线 4

[1,5,9,13……253]对应中断线 1 /HT HI 对应中断线 5

[2,6,10,14……254]对应中断线 2 /HT HI 对应中断线 6

[3,7,11,15……255]对应中断线 3 /HT HI 对应中断线 7

以下中断向量的描述对应于 ht_int_stripe_1，另外两种方式可由以上说明得到。

偏移量： 0x80
复位值： 0x00000000
名称： HT 总线中断向量寄存器[31:0]

表 14-20 HT 总线中断向量寄存器定义（1）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [31:0]	32	0x0	R/W	HT 总线中断向量寄存器[31:0]，对应中断线 0 /HT HI 对应中断线 4

偏移量： 0x84
复位值： 0x00000000
名称： HT 总线中断向量寄存器[63:32]

表 14-21 HT 总线中断向量寄存器定义（2）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [63:32]	32	0x0	R/W	HT 总线中断向量寄存器[63:32]，对应中断线 0 /HT HI 对应中断线 4

偏移量： 0x88
复位值： 0x00000000
名称： HT 总线中断向量寄存器[95:64]

表 14-22 HT 总线中断向量寄存器定义（3）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [95:64]	32	0x0	R/W	HT 总线中断向量寄存器[95:64]，对应中断线 1 /HT HI 对应中断线 5

偏移量： 0x8c
复位值： 0x00000000
名称： HT 总线中断向量寄存器[127:96]

表 14-23 HT 总线中断向量寄存器定义（4）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [127:96]	32	0x0	R/W	HT 总线中断向量寄存器[127:96]，对应中断线 1 /HT HI 对应中断线 5

偏移量： 0x90
复位值： 0x00000000
名称： HT 总线中断向量寄存器[159:128]

表 14-31 HT 总线中断向量寄存器定义（5）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [159:128]	32	0x0	R/W	HT 总线中断向量寄存器[159:128]，对应中断线 2 /HT HI 对应中断线 6

偏移量： 0x94
复位值： 0x00000000
名称： HT 总线中断向量寄存器[191:160]

表 14-24 HT 总线中断向量寄存器定义（6）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [191:160]	32	0x0	R/W	HT 总线中断向量寄存器[191:160]，对应中断线 2 /HT HI 对应中断线 6

偏移量： 0x98
复位值： 0x00000000
名称： HT 总线中断向量寄存器[223:192]

表 14-25 HT 总线中断向量寄存器定义（7）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [223:192]	32	0x0	R/W	HT 总线中断向量寄存器[223:192]，对应中断线 3 /HT HI 对应中断线 7

偏移量： 0x9c
复位值： 0x00000000
名称： HT 总线中断向量寄存器[255:224]

表 14-26 HT 总线中断向量寄存器定义（8）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_case [255:224]	32	0x0	R/W	HT 总线中断向量寄存器[255:224]，对应中断线 3 /HT HI 对应中断线 7

14.5.8 中断使能寄存器

中断使能寄存器共 256 个，与中断向量寄存器一一对应。 置 1 为对应中断打开，置 0 则为中断屏蔽。

256 个中断向量根据中断路由方式选择寄存器配置的不同映射到不同的中断线上，具体的映射方式为：

ht_int_stripe_1:

[0,1,2,3……63]对应中断线 0 /HT HI 对应中断线 4

[64,65,66,67……127]对应中断线 1 /HT HI 对应中断线 5

[128,129,130,131……191]对应中断线 2 /HT HI 对应中断线 6

[192,193,194,195……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_2:

[0,2,4,6……126]对应中断线 0 /HT HI 对应中断线 4

[1,3,5,7……127]对应中断线 1 /HT HI 对应中断线 5

[128,130,132,134……254]对应中断线 2 /HT HI 对应中断线 6

[129,131,133,135……255]对应中断线 3 /HT HI 对应中断线 7

ht_int_stripe_4:

[0,4,8,12……252]对应中断线 0 /HT HI 对应中断线 4

[1,5,9,13……253]对应中断线 1 /HT HI 对应中断线 5

[2,6,10,14……254]对应中断线 2 /HT HI 对应中断线 6

[3,7,11,15……255]对应中断线 3 /HT HI 对应中断线 7

以下中断向量的描述对应于 ht_int_stripe_1，另外两种方式可由以上说明得到。

偏移量： 0xa0
复位值： 0x00000000
名称： HT 总线中断使能寄存器[31:0]

表 14-27 HT 总线中断使能寄存器定义（1）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [31:0]	32	0x0	R/W	HT 总线中断使能寄存器[31:0]， 对应中断线 0 /HT HI 对应中断线 4

偏移量：　　0xa4
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[63:32]

表 14-28 HT 总线中断使能寄存器定义（2）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [63:32]	32	0x0	R/W	HT 总线中断使能寄存器[63:32]， 对应中断线 0 /HT HI 对应中断线 4

偏移量：　　0xa8
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[95:64]

表 14-29 HT 总线中断使能寄存器定义（3）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [95:64]	32	0x0	R/W	HT 总线中断使能寄存器[95:64]， 对应中断线 1 /HT HI 对应中断线 5

偏移量：　　0xac
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[127:96]

表 14-30 HT 总线中断使能寄存器定义（4）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [127:96]	32	0x0	R/W	HT 总线中断使能寄存器[127:96]， 对应中断线 1 /HT HI 对应中断线 5

偏移量：　　0xb0
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[159:128]

表 14-31 HT 总线中断使能寄存器定义（5）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [159:128]	32	0x0	R/W	HT 总线中断使能寄存器[159:128]， 对应中断线 2 /HT HI 对应中断线 6

偏移量：　　0xb4
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[191:160]

表 14-32 HT 总线中断使能寄存器定义（6）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [191:160]	32	0x0	R/W	HT 总线中断使能寄存器[191:160]，对应中断线 2 /HT HI 对应中断线 6

偏移量：　　0xb8
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[223:192]

表 14-33 HT 总线中断使能寄存器定义（7）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [223:192]	32	0x0	R/W	HT 总线中断使能寄存器[223:192]，对应中断线 3 /HT HI 对应中断线 7

偏移量：　　0xbc
复位值：　　0x00000000
名称：　　HT 总线中断使能寄存器[255:224]

表 14-34 HT 总线中断使能寄存器定义（8）

位域	位域名称	位宽	复位值	访问	描述
31:0	Interrupt_mask [255:224]	32	0x0	R/W	HT 总线中断使能寄存器[255:224]，对应中断线 3 /HT HI 对应中断线 7

14.5.9 Link Train 寄存器

HyperTransport 3.0 链路初始化及链路训练控制寄存器。

偏移量：　　0xD0
复位值：　　0x00000070
名称：　　Link Train 寄存器

表 14-35 Link Train 寄存器

位域	位域名称	位宽	复位值	访问	描述
31:23	Reserved	9	0x0	R	保留
22：21	Transmitter LS select	2	0x0	R/W	发送端在 Disconnected 或 Inactive 状态下的链路状态： 2'b00 LS1 2'b01 LS0 2'b10 LS2 2'b11 LS3
14	Dsiable Cmd Throttling	1	0x0	R/W	在 HyperTransport 3.0 模式下，默认任意 4 个连续的 DWS 中只能出现一个 Non-info CMD： 1'b0 使能 Cmd Throttling 1'b1 禁用 Cmd Throttling
13:10	Reserved	4	0x0	R	保留
8：7	Receiver LS select	2	0x0	R/W	接收端在 Disconnected 或 Inactive 状态下的链

					路状态： 2'b00 LS1 2'b01 LS0 2'b10 LS2 2'b11 LS3
6:4	Long Retry Count	3	0x7	R/W	Long Retry 最大次数
3	Scrambling Enable	1	0x0	R/W	是否使能 Scramble 0: 禁用 Scramble 1: 使能 Scramble
2	8B10B Enable	1	0x0	R/W	是否使能 8B10B 0: 禁用 8B10B 1: 使能 8B10B
1	AC	1	0x0	R	是否检测到 AC mode 0: 没有检测到 AC mode 1: 检测到 AC mode
0	Reserved	1	0x0	R	保留

14.5.10 接收地址窗口配置寄存器

HT 控制器中的地址窗口命中公式如下：

```
hit = ( BASE & MASK ) == ( ADDR & MASK )
addr_out_trans = TRANS_EN ? TRANS | ADDR & ~MASK : ADDR
addr_out = Multi_node_en ?
        addr_out_trans[39:37],addr_out_trans[43:40],3' b0,addr_out[36:0]:
        addr_out_trans;
```

需要说明的是，配置地址窗口寄存器时，MASK 高位应全为 1，低位应全为 0。MASK 中 0 的实际位数表示的就是地址窗口的大小。

接收地址窗口的地址为 HT 总线上接收的地址。落在 P2P 窗口内的 HT 地址将作为 P2P 命令转发回 HT 总线，落在正常接收窗口内且不在 P2P 窗口内的 HT 地址将被发往 CPU 内，其它地址的命令将作为 P2P 命令被转发回 HT 总线。

偏移量： 0x140
复位值： 0x00000000
名称： HT 总线接收地址窗口 0 使能（外部访问）

表 14-36 HT 总线接收地址窗口 0 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_image0_en	1	0x0	R/W	HT 总线接收地址窗口 0，使能信号
30	ht_rx_image0_trans_en	1	0x0	R/W	HT 总线接收地址窗口 0，映射使能信号

29	ht_rx_image0_multi_node_en	1	0x0	R/W	HT 总线接收地址窗口 0，多结点地址映射使能 将地址的[39:37]转换到[46:44]
28	ht_rx_image0_conf_hit_en	1	0x0	R/W	HT 总线接收地址窗口 0，协议地址命中使能 必须置 0
25:0	ht_rx_image0_trans[49:24]	26	0x0	R/W	HT 总线接收地址窗口 0，映射后地址的[49:24]

偏移量： 0x144
复位值： 0x00000000
名称： HT 总线接收地址窗口 0 基址（外部访问）

表 14-37 HT 总线接收地址窗口 0 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_image0_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 0，地址基址的[39:24]
15:0	ht_rx_image0_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 0，地址屏蔽的[39:24]

偏移量： 0x148
复位值： 0x00000000
名称： HT 总线接收地址窗口 1 使能（外部访问）

表 14-38 HT 总线接收地址窗口 1 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_image1_en	1	0x0	R/W	HT 总线接收地址窗口 1，使能信号
30	ht_rx_image1_trans_en	1	0x0	R/W	HT 总线接收地址窗口 1，映射使能信号
29	ht_rx_image1_multi_node_en	1	0x0	R/W	HT 总线接收地址窗口 1，多结点地址映射使能 将地址的[39:37]转换到[46:44]
28	ht_rx_image1_conf_hit_en	1	0x0	R/W	HT 总线接收地址窗口 1，协议地址命中使能 必须置 0
25:0	ht_rx_image1_trans[49:24]	26	0x0	R/W	HT 总线接收地址窗口 1，映射后地址的[49:24]

偏移量： 0x14c
复位值： 0x00000000
名称： HT 总线接收地址窗口 1 基址（外部访问）

表 14-39 HT 总线接收地址窗口 1 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_image1_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 1，地址基址的[39:24]
15:0	ht_rx_image1_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 1，地址屏蔽的[39:24]

偏移量： 0x150

复位值： 0x00000000
名称： HT 总线接收地址窗口 2 使能（外部访问）

表 14-40 HT 总线接收地址窗口 2 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_image2_en	1	0x0	R/W	HT 总线接收地址窗口 2，使能信号
30	ht_rx_image2_trans_en	1	0x0	R/W	HT 总线接收地址窗口 2，映射使能信号
29	ht_rx_image2_multi_node_en	1	0x0	R/W	HT 总线接收地址窗口 2，多结点地址映射使能 将地址的[39:37]转换到[46:44]
28	ht_rx_image2_conf_hit_en	1	0x0	R/W	HT 总线接收地址窗口 2，协议地址命中使能 必须置 0
25:0	ht_rx_image2_trans[49:24]	26	0x0	R/W	HT 总线接收地址窗口 2，映射后地址的[49:24]

偏移量： 0x154
复位值： 0x00000000
名称： HT 总线接收地址窗口 2 基址（外部访问）

表 14-41 HT 总线接收地址窗口 2 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_image2_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 2，地址基址的[39:24]
15:0	ht_rx_image2_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 2，地址屏蔽的[39:24]

偏移量： 0x158
复位值： 0x00000000
名称： HT 总线接收地址窗口 3 使能（外部访问）

表 14-42 HT 总线接收地址窗口 3 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_image3_en	1	0x0	R/W	HT 总线接收地址窗口 3，使能信号
30	ht_rx_image3_trans_en	1	0x0	R/W	HT 总线接收地址窗口 3，映射使能信号
29	ht_rx_image3_multi_node_en	1	0x0	R/W	HT 总线接收地址窗口 3，多结点地址映射使能 将地址的[39:37]转换到[46:44]
28	ht_rx_image3_conf_hit_en	1	0x0	R/W	HT 总线接收地址窗口 3，协议地址命中使能 必须置 0
25:0	ht_rx_image3_trans[49:24]	26	0x0	R/W	HT 总线接收地址窗口 3，映射后地址的[49:24]

偏移量： 0x15C
复位值： 0x00000000
名称： HT 总线接收地址窗口 3 基址（外部访问）

表 14-43 HT 总线接收地址窗口 3 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_image3_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 3，地址基址的[39:24]
15:0	ht_rx_image3_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 3，地址屏蔽的[39:24]

偏移量： 0x160
复位值： 0x00000000
名称： HT 总线接收地址窗口 4 使能（外部访问）

表 14-44 HT 总线接收地址窗口 4 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_image4_en	1	0x0	R/W	HT 总线接收地址窗口 4，使能信号
30	ht_rx_image4_trans_en	1	0x0	R/W	HT 总线接收地址窗口 4，映射使能信号
29	ht_rx_image4_multi_node_en	1	0x0	R/W	HT 总线接收地址窗口 4，多结点地址映射使能 将地址的[39:37]转换到[46:44]
28	ht_rx_image4_conf_hit_en	1	0x0	R/W	HT 总线接收地址窗口 4，协议地址命中使能 必须置 0
25:0	ht_rx_image4_trans[49:24]	26	0x0	R/W	HT 总线接收地址窗口 4，映射后地址的[49:24]

偏移量： 0x164
复位值： 0x00000000
名称： HT 总线接收地址窗口 4 基址（外部访问）

表 14-45 HT 总线接收地址窗口 4 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_image4_base[39:24]	16	0x0	R/W	HT 总线接收地址窗口 4，地址基址的[39:24]
15:0	ht_rx_image4_mask[39:24]	16	0x0	R/W	HT 总线接收地址窗口 4，地址屏蔽的[39:24]

14. 5. 11 配置空间转换寄存器

用于对 HT 的配置空间进行各种转换。

偏移量： 0x168
复位值： 0x00000000
名称： 配置空间扩展地址转换

表 14-46 配置空间扩展地址转换寄存器定义

位域	位域名称	位宽	复位值	访问	描述

31	ht_rx_header_trans_ext	1	0x1	R/W	将配置空间（0xFD_FE000000）转换后的地址 type1 标志位由 24 位调整到 28 位，用于与 EXT HEADER 空间统一
30	ht_rx_header_trans_en	1	0x1	R/W	使能配置空间（0xFD_FE000000）的高位地址（[39:24]）转换
29:0	ht_rx_header_trans[53:24]	30	0xFE00	R/W	配置空间转换后的高地址[53:24]（实际只有[53:25]可用）

偏移量： 0x16C
复位值： 0x00000000
名称： 扩展地址转换

表 14-47 扩展地址转换寄存器定义

位域	位域名称	位宽	复位值	访问	描述
30	ht_rx_ext_header_trans_en	1	0x0	R/W	使能扩展配置空间（0xFE_00000000）的高位地址（[39:28]）转换
29:0	ht_rx_ext_header_trans[53:24]	30	0x0	R/W	扩展配置空间转换后的高地址[53:24]（实际只有[53:29]可用）

14.5.12 POST 地址窗口配置寄存器

地址窗口命中公式详见 14.5.10 节。

本窗口的地址是 AXI 总线上接收到的地址。落在本窗口的所有写访问将立即在 AXI B 通道返回，并以 POST WRITE 的命令格式发给 HT 总线。而不在本窗口的写请求则以 NONPOST WRITE 的方式发送到 HT 总线，并等待 HT 总线响应后再返回 AXI 总线。

偏移量： 0x170
复位值： 0x00000000
名称： HT 总线 POST 地址窗口 0 使能（内部访问）

表 14-48 HT 总线 POST 地址窗口 0 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_post0_en	1	0x0	R/W	HT 总线 POST 地址窗口 0，使能信号
30	ht_split0_en	1	0x0	R/W	HT 访问拆包使能(对应于 CPU 核的对外 uncache ACC 操作窗口)
29:23	Reserved	14	0x0		保留
15:0	ht_post0_trans[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 0，转译后地址的[39:24]

偏移量：　　0x174
复位值：　　0x00000000
名称：　　HT 总线 POST 地址窗口 0 基址（内部访问）

表 14-49 HT 总线 POST 地址窗口 0 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_post0_base[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 0，地址基址的[39:24]
15:0	ht_post0_mask[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 0，地址屏蔽的[39:24]

偏移量：　　0x178
复位值：　　0x00000000
名称：　　HT 总线 POST 地址窗口 1 使能（内部访问）

表 14-50 HT 总线 POST 地址窗口 1 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_post1_en	1	0x0	R/W	HT 总线 POST 地址窗口 1，使能信号
30	ht_split1_en	1	0x0	R/W	HT 访问拆包使能(对应于 CPU 核的对外 uncache ACC 操作窗口)
29:16	Reserved	14	0x0		保留
15:0	ht_post1_trans[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 1，转译后地址的[39:24]

偏移量：　　0x17c
复位值：　　0x00000000
名称：　　HT 总线 POST 地址窗口 1 基址（内部访问）

表 14-51 HT 总线 POST 地址窗口 1 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_post1_base[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 1，地址基址的[39:24]
15:0	ht_post1_mask[39:24]	16	0x0	R/W	HT 总线 POST 地址窗口 1，地址屏蔽的[39:24]

14.5.13 可预取地址窗口配置寄存器

地址窗口命中公式详见 14.5.10 节。

本窗口的地址是 AXI 总线上接收到的地址。落在本窗口的取指指令以及 CACHE 访问才会被发往 HT 总线，其它的取指或 CACHE 访问将不会被发往 HT 总线，而是立即返回，如果是读命令，则会返回相应个数的无效读数据。

偏移量：　　0x180

复位值： 0x00000000
名称： HT 总线可预取地址窗口 0 使能（内部访问）

表 14-52 HT 总线可预取地址窗口 0 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_prefetch0_en	1	0x0	R/W	HT 总线可预取地址窗口 0，使能信号
30:16	Reserved	15	0x0		保留
15:0	ht_prefetch0_trans[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 0，转译后地址的[39:24]

偏移量： 0x184
复位值： 0x00000000
名称： HT 总线可预取地址窗口 0 基址（内部访问）

表 14-53 HT 总线可预取地址窗口 0 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_prefetch0_base[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 0，地址基址的[39:24]位地址
15:0	ht_prefetch0_mask[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 0，地址屏蔽的[39:24]

偏移量： 0x188
复位值： 0x00000000
名称： HT 总线可预取地址窗口 1 使能（内部访问）

表 14-54 HT 总线可预取地址窗口 1 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_prefetch1_en	1	0x0	R/W	HT 总线可预取地址窗口 1，使能信号
30:16	Reserved	15	0x0		保留
15:0	ht_prefetch1_trans[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 1，转译后地址的[39:24]

偏移量： 0x18c
复位值： 0x00000000
名称： HT 总线可预取地址窗口 1 基址（内部访问）

表 14-55 HT 总线可预取地址窗口 1 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_prefetch1_base[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 1，地址基址的[39:24]
15:0	ht_prefetch1_mask[39:24]	16	0x0	R/W	HT 总线可预取地址窗口 1，地址屏蔽的[39:24]

14.5.14 UNCACHE 地址窗口配置寄存器

地址窗口命中公式详见 14.5.10 节。

本窗口的地址是 HT 总线上接收到的地址。落在本窗口地址的读写命令，将不会被送往 SCACHE，也不会使一级 CACHE 发生失效，而是会被直接送至内存或是其它的地址空间，也即该地址窗口中的读写命令将不会维持 IO 的 CACHE 一致性。该窗口主要针对一些不会在 CACHE 中命中所以可以提高访存效率的操作，如显存的访问等。

偏移量： 0x190
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 0 使能（内部访问）

表 14-56 HT 总线 Uncache 地址窗口 0 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_uncache0_en	1	0x0	R/W	HT 总线 uncache 地址窗口 0，使能信号
30	ht_uncache0_trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 0，映射使能信号
29	ht_uncache0_multi_node_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 0，多结点地址映射使能
28	ht_uncache0_conf_hit_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 0，协议地址命中使能
25:0	ht_uncache0_trans[49:24]	26	0x0	R/W	HT 总线 uncache 地址窗口 0，转译后地址的[49:24]

偏移量： 0x194
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 0 基址（内部访问）

表 14-57 HT 总线 Uncache 地址窗口 0 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_uncache0_base[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 0，地址基址的[39:24]
15:0	ht_uncache0_mask[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 0，地址屏蔽的[39:24]

偏移量： 0x198
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 1 使能（内部访问）

表 14-58 HT 总线 Uncache 地址窗口 1 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_uncache1_en	1	0x0	R/W	HT 总线 uncache 地址窗口 1，使能信号
30	ht_uncache1_trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 1，映射使能信号
29	ht_uncache1_multi_node_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 1，多结点地址映射使能
28	ht_uncache1_conf_hit_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 1，协议地址命中使能
25:0	ht_uncache1_trans[49:24]	26	0x0	R/W	HT 总线 uncache 地址窗口 1，转译后地址的[49:24]

偏移量：0x19c
复位值：0x00000000
名称：HT 总线 Uncache 地址窗口 1 基址（内部访问）

表 14-59 HT 总线 Uncache 地址窗口 1 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_uncache1_base[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 1，地址基址的[39:24]
15:0	ht_uncache1_mask[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 1，地址屏蔽的[39:24]

偏移量：0x1A0
复位值：0x00000000
名称：HT 总线 Uncache 地址窗口 2 使能（内部访问）

表 14-60 HT 总线 Uncache 地址窗口 2 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_uncache2_en	1	0x0	R/W	HT 总线 uncache 地址窗口 2，使能信号
30	ht_uncache2_trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 2，映射使能信号
29	ht_uncache2_multi_node_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 2，多结点地址映射使能
28	ht_uncache2_conf_hit_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 2，协议地址命中使能
25:0	ht_uncache2_trans[49:24]	26	0x0	R/W	HT 总线 uncache 地址窗口 2，转译后地址的[49:24]

偏移量： 0x1A4
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 2 基址（内部访问）

表 14-61 HT 总线 Uncache 地址窗口 2 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_uncache2_base[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 2，地址基址的[39:24]
15:0	ht_uncache2_mask[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 2，地址屏蔽的[39:24]

偏移量： 0x1A8
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 3 使能（内部访问）

表 14-62 HT 总线 Uncache 地址窗口 3 使能（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31	ht_uncache3_en	1	0x0	R/W	HT 总线 uncache 地址窗口 3，使能信号
30	ht_uncache3_trans_en	1	0x0	R/W	HT 总线 uncache 地址窗口 3，映射使能信号
29	ht_uncache3_multi_node_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 3，多结点地址映射使能
28	ht_uncache3_conf_hit_en	1	0x0	R/W	HT 总线 uncache 接收地址窗口 3，协议地址命中使能
25:0	ht_uncache3_trans[49:24]	26	0x0	R/W	HT 总线 uncache 地址窗口 3，转译后地址的[49:24]

偏移量： 0x1AC
复位值： 0x00000000
名称： HT 总线 Uncache 地址窗口 3 基址（内部访问）

表 14-63 HT 总线 Uncache 地址窗口 3 基址（内部访问）

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_uncache3_base[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 3，地址基址的[39:24]
15:0	ht_uncache3_mask[39:24]	16	0x0	R/W	HT 总线 uncache 地址窗口 3，地址屏蔽的[39:24]

14.5.15 P2P 地址窗口配置寄存器

地址窗口命中公式详见 14.5.10 节。

本窗口的地址是 HT 总线上接收到的地址。落在本窗口地址的读写命令，直接作为 P2P 命令转发回总线，相对于正常接收窗口和 Uncache 窗口，该窗口具有最高优先级。

偏移量： 0x1B0
复位值： 0x00000000
名称： HT 总线 P2P 地址窗口 0 使能（外部访问）

表 14-64 HT 总线 P2P 地址窗口 0 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_p2p0_en	1	0x0	R/W	HT 总线 P2P 地址窗口 0，使能信号
29:0	ht_rx_p2p0_trans[53:24]	30	0x0	R/W	HT 总线 P2P 地址窗口 0，转译后地址的[53:24]

偏移量： 0x1B4
复位值： 0x00000000
名称： HT 总线 P2P 地址窗口 0 基址（外部访问）

表 14-65 HT 总线 P2P 地址窗口 0 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_p2p0_base[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址基址的[39:24]
15:0	ht_rx_p2p0_mask[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址屏蔽的[39:24]

偏移量： 0x1B8
复位值： 0x00000000
名称： HT 总线 P2P 地址窗口 1 使能（外部访问）

表 14-66 HT 总线 P2P 地址窗口 1 使能（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31	ht_rx_p2p1_en	1	0x0	R/W	HT 总线 P2P 地址窗口 1，使能信号
29:0	ht_rx_p2p1_trans[53:24]	30	0x0	R/W	HT 总线 P2P 地址窗口 1，转译后地址的[53:24]

偏移量： 0x1BC
复位值： 0x00000000
名称： HT 总线 P2P 地址窗口 1 基址（外部访问）

表 14-67 HT 总线 P2P 地址窗口 1 基址（外部访问）寄存器定义

位域	位域名称	位宽	复位值	访问	描述
31:16	ht_rx_p2p1_base[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址基址的[39:24]
15:0	ht_rx_p2p1_mask[39:24]	16	0x0	R/W	HT 总线 P2P 地址窗口 1，地址屏蔽的[39:24]

14.5.16 控制器参数配置寄存器

偏移量： 0x1C0

复位值： 0x00904321
名称： APP CONFIG 0

表 14-68 控制器参数配置寄存器 0 定义

位域	位域名称	位宽	复位值	访问	描述
31:30	Reserved	1	0x0		保留
29	Ldt Stop Gen	1	0x0	R/W	使总线进入 LDT DISCONNECT 模式 正确的方法是：0 -> 1
28	Ldt Req Gen	1	0x0	R/W	从 LDT DISCONNECT 中唤醒 HT 总线，设置 LDT_REQ_n 正确的方法是先置 0 再置 1：0 -> 1 除此之外，直接向总线发出读写请求也可以自动唤醒总线
27	rx sample en	1	0x0	R/W	使能采样输入的 cad 和 ctl，在（0x1c8）寄存器中显示，用于调试
26	Dword Write	1	0x1	R/W	对于 32/64/128/256 位 MEM 写访问，是否采用 Dword Write 命令格式（Byte Write 方式的写在接收时会转换为 128 位带 MASK 的写）
25	Dword Write cfg	1	0x1	R/W	对于配置空间的写访问，是否采用 Dword Write 命令格式（Byte Write 方式的写在接收时会转换为 128 位带 MASK 的写）
24	Dword Write IO	1	0x1	R/W	对于 IO 空间的写访问，是否采用 Dword Write 命令格式（Byte Write 方式的写在接收时会转换为 128 位带 MASK 的写）
23	axi aw resize	1	0x0	RW	是否对 128 位带 MASK 写按 Mask 进行 size 的重新设置
22	Coherent Mode	1	0x0	RW	是否是处理器一致性模式，初始值由 ICCC_EN 引脚决定，复位后生效
21	Coherent_split	1	0x0	RW	一致性模式下，将所有包拆为 32byte 处理
20	Not Care Seqid	1	0x0	R/W	接收端是否不关心 HT 序关系
19:16	Fixed Seqid	4	0x0	R/W	当 Not Axi2Seqid 有效时，配置 HT 总线发出的 Seqid
15:12	Priority Nop	4	0x4	R/W	HT 总线 Nop 流控包优先级
11:8	Priority NPC	4	0x3	R/W	Non Post 通道读写优先级
7:4	Priority RC	4	0x2	R/W	Response 通道读写先级
3:0	Priority PC	4	0x1	R/W	Post 通道读写优先级 0x0：最高优先级 0xF：最低优先级 对于各个通道的优先级均采用根据时间变化提高的优先级策略，该组存器用于配置各个通道的初始优先级

偏移量： 0x1C4

复位值： 0x00904321
名称： APP CONFIG1

表 14-69 控制器参数配置寄存器 1 定义

位域	位域名称	位宽	复位值	访问	描述
31	tx post split en	1	0x0	R/W	使能 tx post ID 窗口命中时的写拆包功能（所有跨越 32 字节边界的写请求会被拆为两个连续的写请求（byte write））
30	tx wr passPW pc	1	0x0	R/W	将所有发出的 Post 通道写请求的 passPW 位设置为 1
29	tx wr passPW npc	1	0x0	R/W	将所有发出的 Nonpost 通道写请求的 passPW 位设置为 1
28	tx rd passPW	1	0x0	R/W	将所有发出的读请求的 passPW 位设置为 1
27	stop same id wr	1	0x0	R/W	发送端遇到相同 AXI ID 的写请求时，停止发送直至前一个同 ID 请求返回
26	stop same id rd	1	0x0	R/W	发送端遇到相同 AXI ID 的读请求时，停止发送直至前一个同 ID 请求返回
25	Not axi2seqid wr	1	0x0	R/W	禁止写请求 AXI ID 到 seqid 的转换，直接使用 fixed seqid
24	Not axi2seqid rd	1	0x0	R/W	禁止读请求 AXI ID 到 seqid 的转换，直接使用 fixed seqid
23:22	Reserved	2	0x0	R/W	保留
21	act as slave	1	0x1	R/W	设置 SLAVE 模式
20	Host hide	1	0x0	R/W	禁止接收端对配置寄存器空间的访问
19:16	Rrequest delay	4	0x3	R/W	用于在一致性模式下，控制 Rrequest 传输的随机延迟范围 000：0 延迟 001：随机延迟 0-8 010：随机延迟 8-15 011：随机延迟 16-31 100：随机延迟 32-63 101：随机延迟 64-127 110：随机延迟 128-255 111：0 延迟
15	Crc Int en	1	0x0	R/W	使能 CRC 错时的中断发送
14:12	Crc Int route	3	0x0	R/W	CRC 中断时的中断引脚选择
11	Reserved				
10	ht int 8 bit	1	0x0	R/W	使用 8 根中断线
9:8	ht_int_stripe	2	0x0	R/W	对应于 3 种中断路由方式，具体描述见 中断向量寄存器 0x0：ht_int_stripe_1 0x1：ht_int_stripe_2

					0x2：ht_int_stripe_4
4:0	Interrupt Index	5	0x0	R/W	将除了标准中断之外的其它中断重定向到哪个中断向量中（包括 SMI，NMI，INIT，INTA，INTB，INTC，INTD） 总共 256 个中断向量，本寄存器表示的是中断向量的高 5 位，内部中断向量如下： 000：SMI 001：NMI 010：INIT 011：Reservered 100：INTA 101：INTB 110：INTC 111：INTD

14.5.17 接收诊断寄存器

偏移量：　　0x1C8
复位值：　　0x00000000
名称：　　接收诊断寄存器

表 14-70 接收诊断寄存器

位域	位域名称	位宽	复位值	访问	描述
31:16	rx_cad_phase_0	16	0x0	R/W	保存采样得到的输入 CAD[15:0]的值
15:8	rx_ctl_catch	8	0x0	R/W	保存采样得到的输入 ctl （0、2、4、6）对应 CTL0 采样的四个相位 （1、3、5、7）对应 CTL1 采样的四个相位
7:0					

14.5.18 PHY 状态寄存器

用于观测 PHY 相关的状态，调试使用

偏移量：　　0x1CC
复位值：　　0x83308000
名称：　　PHY 状态寄存器

表 14-71 PHY 状态寄存器

位域	位域名称	位宽	复位值	访问	描述
31:29	Reserved	3	0x0	R	保留
28	dll locked hi	1	0x0	R	高 8 位 DLL 锁定
27	dll locked lo	1	0x0	R	低 8 位 DLL 锁定

26	cdr locked hi	1	0x0	R	高 8 位 CDR 锁定
25	cdr locked lo	1	0x0	R	低 8 位 CDR 锁定
24	phase locked	1	0x0	R	相位锁定
23:20	phy state	4	0x0	R	PHY 状态
19:17	tx training status	3	0x0	R	TX 训练状态
16:14	rx training status	3	0x0	R	RX 训练状态
13:8	Init done	6	0x0	R	初始化完成
7:0	Reserved	8		R	保留

14.5.19 命令发送缓存大小寄存器

命令发送缓存大小寄存器用于观测发送端可用各个命令通道的缓存个数。

偏移量： 0x1D0
复位值： 0x00000000
名称： 命令发送缓存大小寄存器

表 14-72 命令发送缓存大小寄存器

位域	位域名称	位宽	复位值	访问	描述
31:24	B_CMD_txbuffer	8	0x0	R	发送端 B 通道命令缓存个数
23:16	R_CMD_txbuffer	8	0x0	R	发送端 R 通道命令缓存个数
15:8	NPC_CMD_txbuffer	8	0x0	R	发送端 NPC 通道命令缓存个数
7:0	PC_CMD_txbuffer	8	0x0	R	发送端 PC 通道命令缓存个数

14.5.20 数据发送缓存大小寄存器

数据发送缓存大小寄存器用于观测发送端可用各个数据通道的缓存个数。

偏移量： 0x1D4
复位值： 0x00000000
名称： 数据发送缓存大小寄存器

表 14-73 数据发送缓存大小寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Reserved	1	0x0	R	保留
30	rx_buffer_r_data[4]	1	0x0	R/W	接收缓冲区的读数据 buffer 初始化信息的 bit[4]
29	rx_buffer_npc_data[4]	1	0x0	R/W	接收缓冲区的 npc 数据 buffer 初始化信息的 bit[4]
28	rx_buffer_pc_data[4]	1	0x0	R/W	接收缓冲区的 pc 数据 buffer 初始化信息的 bit[4]

27	rx_buffer_b_cmd[4]	1	0x0	R/W	接收缓冲区的 bresponse 命令 buffer 初始化信息的 bit[4]
26	rx_buffer_r_cmd[4]	1	0x0	R/W	接收缓冲区的读命令 buffer 初始化信息的 bit[4]
25	rx_buffer_npc_cmd[4]	1	0x0	R/W	接收缓冲区的 npc 命令 buffer 初始化信息的 bit[4]
24	rx_buffer_pc_cmd[4]	1	0x0	R/W	接收缓冲区的 pc 命令 buffer 初始化信息的 bit[4]
23:16	R_DATA_txbuffer	8	0x0	R	发送端 R 通道数据缓存个数
15:8	NPC_DATA_txbuffer	8	0x0	R	发送端 NPC 通道数据缓存个数
7:0	PC_DATA_txbuffer	8	0x0	R	发送端 PC 通道数据缓存个数

14.5.21 发送缓存调试寄存器

发送缓存调试寄存器用于人工设置 HT 控制器发送端缓冲区的个数，通过增或减的方式对不同的发送缓存个数进行调整。

偏移量： 0x1D8
复位值： 0x00000000
名称： 发送缓存调试寄存器

表 14-74 发送缓存调试寄存器

位域	位域名称	位宽	复位值	访问	描述
31	b_interleave	1	0x0	R/W	一致性模式下，使能 B 通道与其它通道的交错传输
30	nop_interleave	1	0x0	R/W	使能流控包与其它虚通道的交错传输
29	Tx_neg	1	0x0	R/W	发送端缓存调试符号 0：增加相应个数 1：减少（相应寄存器个数+1）个
28	Tx_buff_adj_en	1	0x0	R/W	发送端缓存调试使能寄存器 0->1：使本寄存器的值产生一次增减效果
27:24	R_DATA_txadj	4	0x0	R/W	发送端 R 通道数据缓存增减个数 当 tx_neg 为 0 时，增加 R_DATA_txadj 个； 当 tx_neg 为 1 时，减少 R_DATA_txadj+1 个
23:20	NPC_DATA_txadj	4	0x0	R/W	发送端 NPC 通道数据缓存增减个数 当 tx_neg 为 0 时，增加 NPC_DATA_txadj 个； 当 tx_neg 为 1 时，减少 NPC_DATA_txadj+1 个
19:16	PC_DATA_txadj	4	0x0	R/W	发送端 PC 通道数据缓存增减个数 当 tx_neg 为 0 时，增加 PC_DATA_txadj 个；

					当 tx_neg 为 1 时，减少 PC_DATA_txadj+1 个
15:12	B_CMD_txadj	4	0x0	R/W	发送端 B 通道命令缓存增减个数 当 tx_neg 为 0 时，增加 B_CMD_txadj 个； 当 tx_neg 为 1 时，减少 B_CMD_txadj+1 个
11:8	R_CMD_txadj	4	0x0	R/W	发送端 R 通道命令缓存增减个数 当 tx_neg 为 0 时，增加 R_CMD_txadj 个； 当 tx_neg 为 1 时，减少 R_CMD_txadj+1 个
7:4	NPC_CMD_txadj	4	0x0	R/W	发送端 NPC 通道命令/数据缓存增减个数 当 tx_neg 为 0 时,增加 NPC_CMD_txadj 个； 当 tx_neg 为 1 时，减少 NPC_CMD_txadj+1 个
3:0	PC_CMD_txadj	4	0x0	R/W	发送端 PC 通道命令缓存增减个数 当 tx_neg 为 0 时，增加 PC_CMD_txadj 个； 当 tx_neg 为 1 时,减少 PC_CMD_txadj+1 个

14.5.22 接收缓冲区初始寄存器

偏移量：　　0x1DC
复位值：　　0x07778888
名称：　　接收缓冲区初始化配置寄存器

表 14-75 接收缓冲区初始寄存器

位域	位域名称	位宽	复位值	访问	描述
27:24	rx_buffer_r_data	4	0x0	R/W	接收缓冲区的读数据 buffer 初始化信息
23:20	rx_buffer_npc_data	4	0x0	R/W	接收缓冲区的 npc 数据 buffer 初始化信息
19:16	rx_buffer_pc_data	4	0x0	R/W	接收缓冲区的 pc 数据 buffer 初始化信息
15:12	rx_buffer_b_cmd	4	0x0	R/W	接收缓冲区的 bresponse 命令 buffer 初始化信息
11:8	rx_buffer_r_cmd	4	0x0	R/W	接收缓冲区的读命令 buffer 初始化信息
7:4	rx_buffer_npc_cmd	4	0x0	R/W	接收缓冲区的 npc 命令 buffer 初始化信息
3:0	rx_buffer_pc_cmd	4	0x0	R/W	接收缓冲区的 pc 命令 buffer 初始化信息

14.5.23 Training 0 超时短计时寄存器

用于配置 HyerTransport 3.0 模式下 Training 0 短计时超时阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量：　　0x1E0
复位值：　　0x00000080

名称： Training 0 超时短计数寄存器

表 14-76 Training 0 超时短计时寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Gen3_timing_soft	1	0x0	R/W	
30:23	Retry_nop_num	8	0x0	R/W	
22:0	T0 time	23	0x80	R/W	Training 0 超时短计时寄存器

14.5.24 Training 0 超时长计时寄存器

用于 HyerTransport 3.0 模式下 Training 0 长计数超时阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x1E4
复位值： 0x000fffff
名称： Training 0 超时长计数寄存器

表 14-77 Training 0 超时长计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T0 time	32	0xfffff	R/W	Training 0 超时长计数寄存器

14.5.25 Training 1 计数寄存器

用于 HyerTransport 3.0 模式下 Training 1 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x1E8
复位值： 0x0004fffff
名称： Training 1 计数寄存器

表 14-78 Training 1 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T1 time	32	0x4fffff	R/W	Training 1 计数寄存器

14.5.26 Training 2 计数寄存器

用于 HyerTransport 3.0 模式下 Training 2 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x1EC

复位值： 0x0007fffff
名称： Training 2 计数寄存器

表 14-79 Training 2 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T2 time	32	0x7fffff	R/W	Training 2 计数寄存器

14.5.27 Training 3 计数寄存器

用于 HyerTransport 3.0 模式下 Training 3 计数阈值，计数器时钟频率为 HyperTransport3.0 链路总线时钟频率的 1/4。

偏移量： 0x1F0
名称： Training 3 计数寄存器

表 14-80 Training 3 计数寄存器

位域	位域名称	位宽	复位值	访问	描述
31:0	T3 time	32	0x7fffff	R/W	Training 3 计数寄存器

14.5.28 软件频率配置寄存器

用于实现控制器在工作过程中切换到任意协议和 PLL 支持的链路频率及控制器频率；

具体切换方法为：在使能软件配置模式的前提下，置位软件频率配置寄存器第 1 位，并写入新的时钟相关的参数，包括决定 PLL 输出频率的 div_refc 和 div_loop ，链路上的分频系数 phy_hi_div 和 phy_lo_div，以及控制器的分频系数 core_div 。之后进入 warm reset 或 LDT disconnect，控制器将会自动复位 PLL，配置新的时钟参数。

PHY_LINK_CLK 为 HT 总线频率。

时钟频率的计算公式为：

当使用 SYS_CLOCK 为参考时钟输入且 SYS_CLOCK 为 25MHz 时（CLKSEL[8]为 1 且 CLKSEL[5]为 1），频率计算方法为：

HyperTransport 1.0:

PHY_LINK_CLK = 12.5MHz×div_loop /div_refc /phy_div

HyperTransport 3.0:

PHY_LINK_CLK = 25MHz×div_loop /div_refc /phy_div

其它情况下，频率计算方法为：

HyperTransport 1.0:

PHY_LINK_CLK ＝ 50MHz×div_loop /div_refc /phy_div

HyperTransport 3.0:

PHY_LINK_CLK ＝ 100MHz×div_loop /div_refc /phy_div

等待 PLL 重新锁定的时间在缺省情况下，system clk 为 33M 时约为 30us；也可以在寄存器中写入自定义的等待计数上限。

需要注意的是，在 3A4000 中，HT_CORE_CLK 不再由这个配置控制，而是由 NODE 时钟分频控制。

偏移量： 0x1F4
复位值： 0x00000000
名称： 软件频率配置寄存器

表 14-81 软件频率配置寄存器

位域	位域名称	位宽	复位值	访问	描述
31：27	PLL relock counter	5	0x0	R/W	计数器上限配置寄存器，当置位 counter select 时，计数器计数上限为{PLL_relock_counter，5’h1f}，否则计数上限为 10’3ff
26	Counter select	1	0x0	R/W	锁定计时器自定义使能： 1’b0 使用默认计数上限； 1’b1 由 PLL_relock_counter 计算得出
25：22	Soft_phy_lo_div	4	0x0	R/W	低位 PHY 分频系数
21：18	Soft_phy_hi_div	4	0x0	R/W	高位 PHY 分频系数
17：16	Soft_div_refc	2	0x0	R/W	PLL 内分频系数
15：9	Soft_div_loop	7	0x0	R/W	PLL 内倍频系数
8：5	Soft_core_div	4	0x0	R/W	控制器时钟分频系数
4：2	Reserved	3	0x0	R	保留
1	Soft cofig enable	1	0x0	R/W	软件配置使能位 1’b0 禁用软件频率配置 1’b1 使能软件频率配置
0	Reserved	1	0x0	R	保留

14.5.29 PHY 阻抗匹配控制寄存器

用于控制 PHY 的阻抗匹配使能，发送端和接收端阻抗匹配参数设置

偏移量： 0x1F8
复位值： 0x00000000
名称： PHY 阻抗匹配控制寄存器

表 14-82 阻抗匹配控制寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Tx_scanin_en	1	0x0	R/W	TX 阻抗匹配使能
30	Rx_scanin_en	1	0x0	R/W	RX 阻抗匹配使能
27:24	Tx_scanin_ncode	4	0x0	R/W	TX 阻抗匹配扫描输入 ncode
23:20	Tx_scanin_pcode	4	0x0	R/W	TX 阻抗匹配扫描输入 pcode
19:12	Rx_scanin_code	8	0x0	R/W	RX 阻抗匹配扫描输入

14.5.30 PHY 配置寄存器

用于配置 PHY 相关的物理参数，当控制器做为两个独立的 8bit 控制器时，高位的 PHY 和低位的 PHY 分别由两个控制器独立控制；当控制器作为 1 个 16bit 的控制器时，高位和低位的 PHY 的配置参数由低位控制器统一控制；

偏移量： 0x1FC
复位值： 0x83308000
名称： PHY 配置寄存器

表 14-83 PHY 配置寄存器

位域	位域名称	位宽	复位值	访问	描述
31	Rx_ckpll_term	1	0x1	R/W	PLL 到 RX 端片上传输线终端阻抗
30	Tx_ckpll_term	1	0x0	R/W	PLL 到 TX 端片上传输线终端阻抗
29	Rx_clk_in_sel_	1	0x0	R/W	时钟 PAD 供给数据 PAD 的时钟选择，HT1 模式下自动选择为 CLKPAD: 1'b0 外来时钟源 1'b1 PLL 时钟
28	Rx_ckdll_sell	1	0x0	R/W	用来锁定 DLL 的时钟选择： 1'b0 PLL 时钟 1'b1 外来时钟源
27:26	Rx_ctle_bitc	2	0x0	R/W	PAD EQD 高频增益

25:24	Rx_ctle_bitr	2	0x3	R/W	PAD EQD 低频增益
23:22	Rx_ctle_bitlim	2	0x0	R/W	PAD EQD 补偿限制
21	Rx_en_ldo	1	0x1	R/W	LDO 控制 1'b0 LDO 禁用 1'b1 LDO 使能
20	Rx_en_by	1	0x1	R/W	BandGap 控制 1'b0 BandGap 禁用 1'b1 BandGap 使能
19：17	Reserved	3	0x0	R	保留
16:12	Tx_preenmp	5	0x08	R/W	PAD 预加重控制信号
11：0	Reserved	12	0x0	R	保留

14.5.31 链路初始化调试寄存器

用于配置在 HyperTransport 3.0 模式下，链路初始化过程中是否使用 PHY 提供的 CDR lock 信号做为链路 CDR 完成的标志；如果忽略该锁定信号，则需要控制器计数等待一定时间后默认 CDR 完成。

偏移量： 0x240
复位值： 0x00000000
名称： 链路初始化调试寄存器

表 14-84 链路初始化调试寄存器

位域	位域名称	位宽	复位值	访问	描述
15	Cdr_ignore_enable	1	0x0	R/W	链路初始化时是否忽略 CRC lock ，通过计数器计数完成等待： 1’b0 等待 CDR lock 1’b1 忽略 CDR lock 信号，通过计数器累加等待
14：0	Cdr_wait_counter	15	0x0	R/W	等待计数器计数上限，基于控制器时钟完成计数

14.5.32 LDT 调试寄存器

软件改变控制器频率后，会导致对 LDT reconnect 阶段计时不准确，需配置该计数器，作为软件配置频率后，LDT 信号无效到控制器开始链路初始化之间的时间,该计时基于控制器时钟。

偏移量： 0x244
复位值： 0x00000000
名称： LDT 调试寄存器 1

表 14-85 LDT 调试寄存器 1

位域	位域名称	位宽	复位值	访问	描述
31:16	Rx_wait_time	16	0x0	R/W	RX 端等待计数器的初值
15:0	Tx_wait_time	16	0x0	R/W	TX 端等待计数器的初值

偏移量： 0x248
复位值： 0x00000000
名称： LDT 调试寄存器 2

表 14-86 LDT 调试寄存器 2

位域	位域名称	位宽	复位值	访问	描述
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 0	16	0x0	R/W	

偏移量： 0x24C
复位值： 0x00000000
名称： LDT 调试寄存器 3

表 14-87 LDT 调试寄存器 3

位域	位域名称	位宽	复位值	访问	描述
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 1	16	0x0	R/W	

偏移量： 0x250
复位值： 0x00000000
名称： LDT 调试寄存器 4

表 14-88 LDT 调试寄存器 4

位域	位域名称	位宽	复位值	访问	描述
31:30	Reserved	16	0x0	R/W	
29:0	rx lane ts 2	16	0x0	R/W	

偏移量： 0x254
复位值： 0x00000000
名称： LDT 调试寄存器 5

表 14-89 LDT 调试寄存器 5

位域	位域名称	位宽	复位值	访问	描述

31:22	Reserved	10	0x0	R/W	
21:18	wait ctl	4	0x0	R/W	
17:0	phase lock	18	0x0	R/W	

偏移量： 0x258
复位值： 0x00000000
名称： LDT 调试寄存器 5

表 14-90 LDT 调试寄存器 5

位域	位域名称	位宽	复位值	访问	描述
31:0	wait cad	32	0x0	R/W	

14.5.33 HT TX POST ID 窗口配置寄存器

该窗口通过将内部写请求的 ID 与预设的窗口相比较，将命中的请求通过 HT POST 通道向外发出。

偏移量： 0x260
复位值： 0x00000000
名称： HT TX POST ID WIN0

表 14-91 HT TX POST ID WIN0

位域	位域名称	位宽	复位值	访问	描述
31:16	HT TX POST ID0 MASK	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 MASK 位
15:0	HT TX POST ID0 BASE	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 BASE 位

偏移量： 0x264
复位值： 0x00000000
名称： HT TX POST ID WIN1

表 14-92 HT TX POST ID WIN1

位域	位域名称	位宽	复位值	访问	描述
31:16	HT TX POST ID1 MASK	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 MASK 位
15:0	HT TX POST ID1 BASE	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 BASE 位

偏移量： 0x268
复位值： 0x00000000

名称： HT TX POST ID WIN2

表 14-93 HT TX POST ID WIN2

位域	位域名称	位宽	复位值	访问	描述
31:16	HT TX POST ID2 MASK	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 MASK 位
15:0	HT TX POST ID2 BASE	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 BASE 位

偏移量： 0x26C
复位值： 0x00000000
名称： HT TX POST ID WIN3

表 14-94 HT TX POST ID WIN3

位域	位域名称	位宽	复位值	访问	描述
31:16	HT TX POST ID3 MASK	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 MASK 位
15:0	HT TX POST ID3 BASE	16	0x0	R/W	AXI ID 命中的请求使用 POST 窗口进行传输，ID 的 BASE 位

14.5.34 外部中断转换配置

该设置将 HT 收到的中断转换为对某个特定地址的写操作，直接写入芯片内部的扩展 IO 中断向量，而不是在 HT 控制器内部产生中断。采用这种方式，可以使用 IO 中断的直接跨片分发等高级功能。

偏移量： 0x270
复位值： 0x00000000
名称： HT RX INT TRANS Lo

表 14-95 HT RX INT TRANS LO

位域	位域名称	位宽	复位值	访问	描述
31:4	INT_trans_addr[31:4]	28	0x0	R/W	中断转换地址低位
3:0	Reserved	4	0x0	R	保留

偏移量： 0x274
复位值： 0x00000000
名称： HT RX INT TRANS Hi

表 14-96 HT RX INT TRANS Hi

位域	位域名称	位宽	复位值	访问	描述
31	INT_trans_en	1	0x0	R/W	中断转换使能
30	INT_trans_allow	1	0x0	R/W	中断转换使能允许 设置该位后，INT_trans_en 或者芯片的 EXT_INT_en 才可生效。
29:26	INT_trans_cache	4	0x0	R/W	中断转换 Cache 域
25:0	INT_trans_addr[57:32]	26	0x0	R/W	中断转换地址高位

14.6 HyperTransport 总线配置空间的访问方法

HyperTransport 接口软件层的协议与 PCI 协议基本一致，由于配置空间的访问直接与底层协议相关，具体访问细节略有不同。在表 14-6 中已列出，HT 总线配置空间的地址范围是 0xFD_FE00_0000 ～0xFD_FFFF_FFFF。对于 HT 协议中的配置访问，在龙芯 3A4000 中采用如下格式实现：

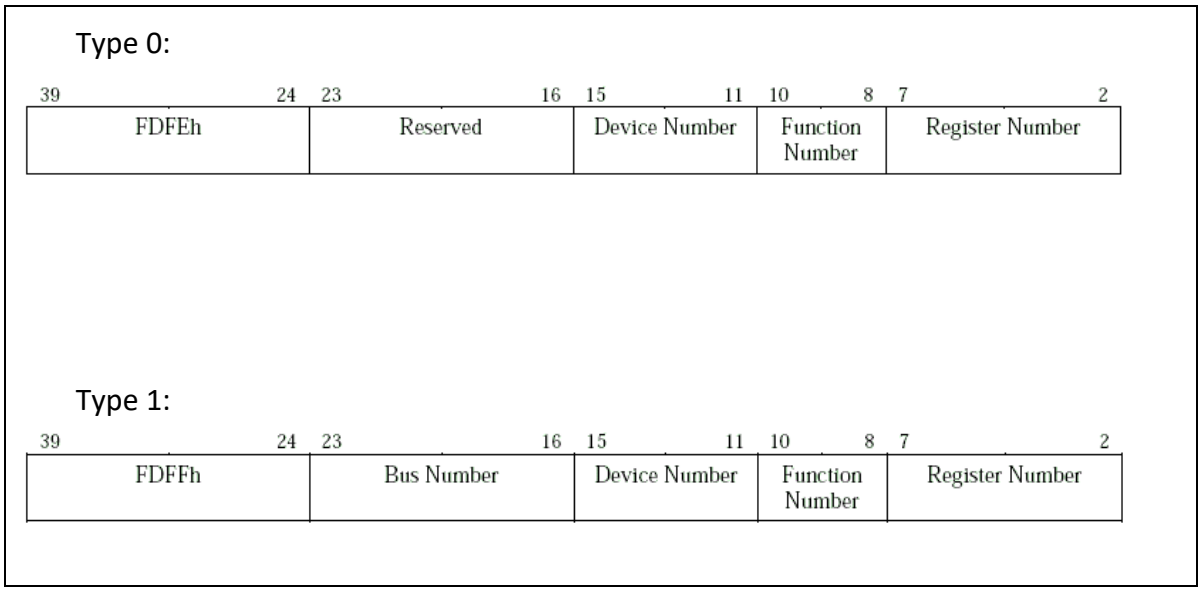


图 14-1 龙芯 3A4000 中 HT 协议的配置访问

14.7 HyperTransport 多处理器支持

龙芯 3 号处理器使用 HyperTransport 接口进行多处理器互连，并且可以硬件自动维护 2-8 个芯片之间的一致性请求。

龙芯 3 号互连路由

龙芯 3 号互连路由有两种方法，一是采用简单 X-Y 路由方法。路由时，先 X 后 Y，以四片芯片为例，ID 号分别为 00，01，10，11。如果从 11 向 00 发出请求，则为 11 向 00 路由，首先走 X 方向，从 11 走到 10，再走 Y 方向，从 10 走到 00。其响应从 00 返回 11 时，路由首先走 X 方向，从 00 到 01，再走 Y 方向，从 01 到 11。另一种是对角线直接访问，通过硬件连接两个对角芯片实现直接访问，大大减少访问延迟，这种访问方式需要通过软件单独使能。由于这个算法的特征，我们在构建多片芯片互连的时候，可以采用多种不同的办法。

四片龙芯 3 号互连结构

四片 CPU 两两相联构成环状结构。每个 CPU 利用 HT0 的两个 8 位控制器与相邻两片相联，利用 HT1 HI 与对角芯片相联，由此而得到下图的互连结构：

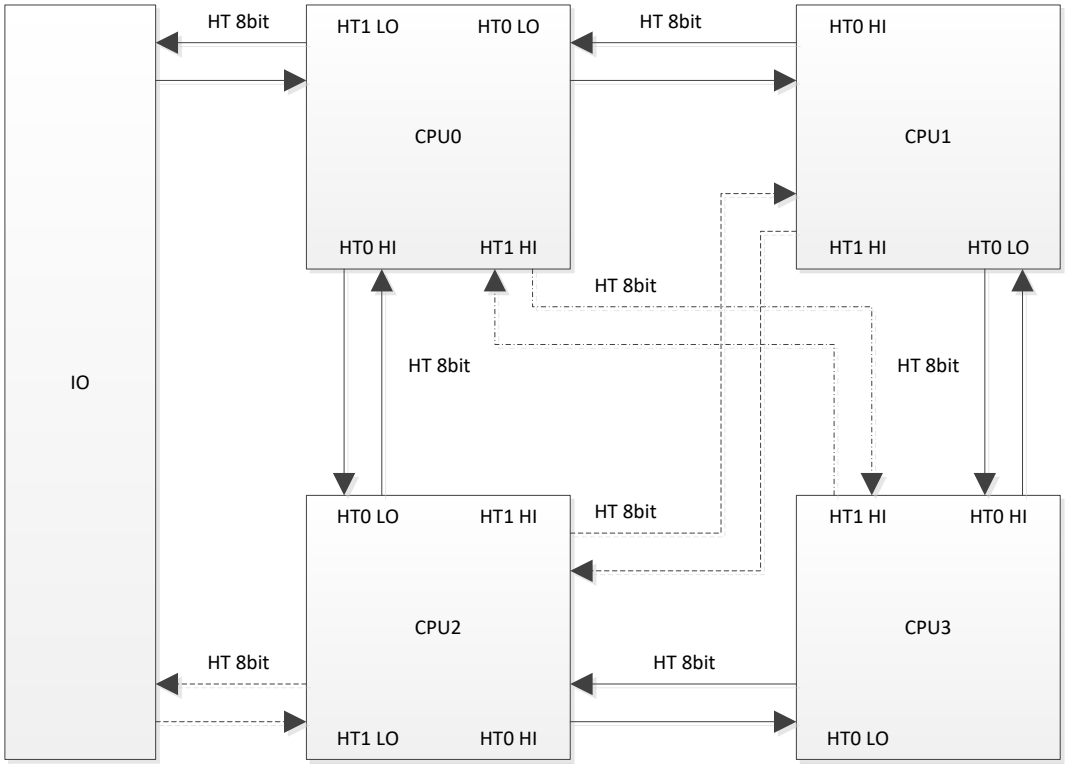


图 14-2 四片龙芯 3 号互连结构

八片龙芯 3 号互连结构

八片 CPU 构成立方体结构。每个 CPU 利用 HT0 的两个 8 位控制器与相邻两片相联，利用 HT1 由此而得到下图的互连结构：

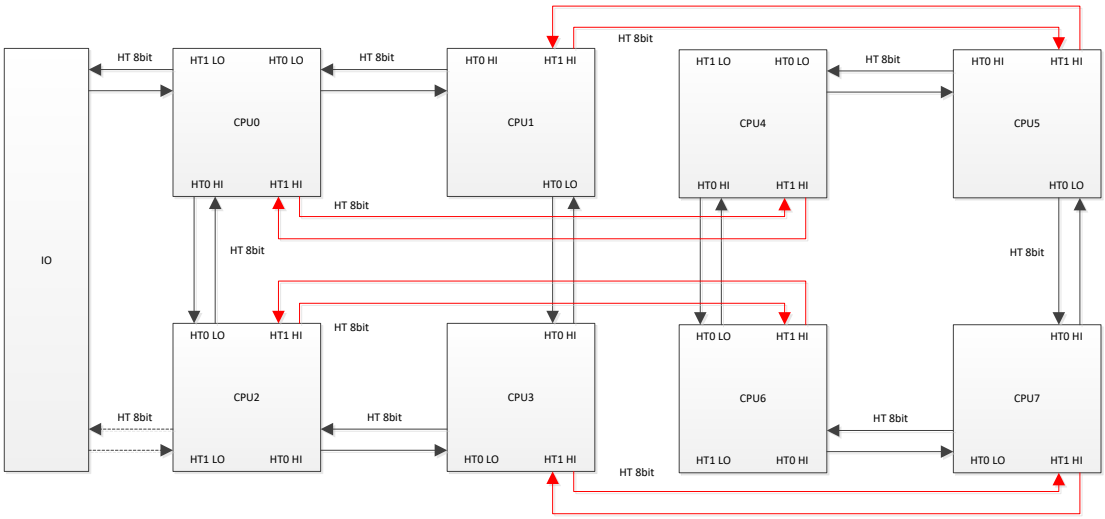


图 14-3 八片龙芯 3 号互连结构

两片龙芯 3 号互连结构

由于固定路由算法的特性，我们在构建两片芯片互连时，有两种不同的方法。首先是采用 8 位 HT 总线互连。这种互连方式下，两个处理器之间只能采用 8 位 HT 互连。两个芯片号分别为 00 与 01，由路由算法，我们可以知道，两个芯片相互访问时都是通过与四片互连时一致的 8 位 HT 总线。如下所示：

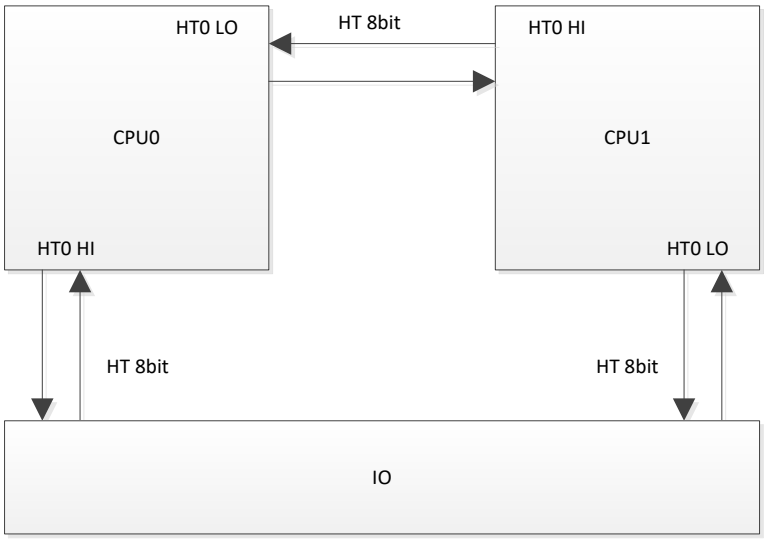


图 14-4 两片龙芯 3 号 8 位互连结构

但是，HT 总线最宽可以采用 16 位模式，由此最大化带宽的连接方式应该是采用 16 位互连结构。在龙芯三号中，只要把 HT0 控制器设置为 16 位模式，所有发到 HT0 控制器的命令都会被发往 HT0_LO，而不是以前的按照路由表分别发至 HT0_HI 或是 HT0_LO，这样，我

们就可以在互连时使用 16 位总线。所以，我们只需要将 CPU0 与 CPU1 的 16 位模式正确配置并将高低位总线正确连接即可使用 16 位 HT 总线互连。而这种互连结构同时也可以使用 8 位的 HT 总线协议进行相互访问。所得到的互连结构如下：

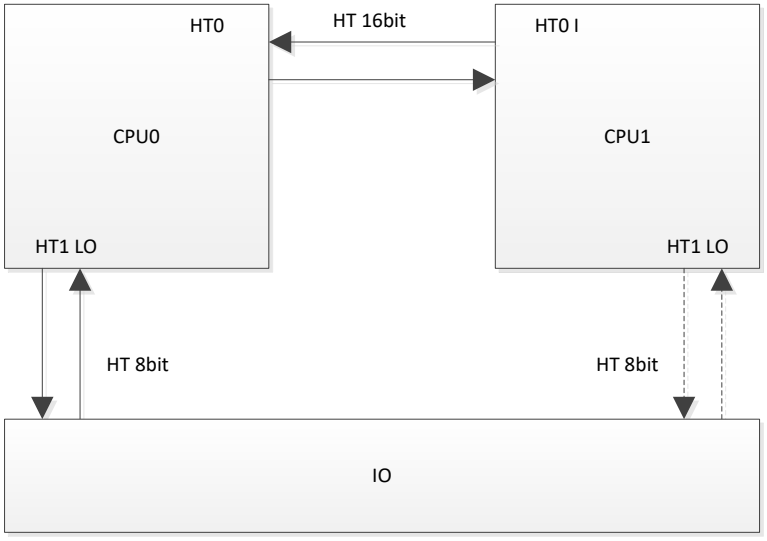


图 14-5 两片龙芯 3 号 16 位互连结构

15 低速 IO 控制器配置

龙芯 3 号 I/O 控制器包括 UART 控制器、SPI 控制器、I2C 及 GPIO 寄存器。这些 I/O 控制器共享一个 AXI 端口，CPU 的请求经过地址译码后发送到相应的设备。

15.1 UART 控制器

UART 控制器具有以下特性

- 全双工异步数据接收/发送
- 可编程的数据格式
- 16 位可编程时钟计数器
- 支持接收超时检测
- 带仲裁的多中断系统
- 仅工作在 FIFO 方式
- 在寄存器与功能上兼容 NS16550A

芯片内部集成两个 UART 接口，功能寄存器完全一样，只是访问基址不同。

UART0 寄存器物理地址基址为 0x1FE001E0。

UART1 寄存器物理地址基址为 0x1FE001E8。

针对这两个 UART 还各提供一个物理地址，分别为 0x1FE00100(UART0)和 0x1FE00110(UART1)。通过这组地址可访问新增的两个寄存器 RFC 和 TFC。

15.1.1 数据寄存器（DAT）

中文名：　　数据传输寄存器
寄存器位宽：　[7：0]
偏移量：　　0x00
复位值：　　0x00

位域	位域名称	位宽	访问	描述
7:0	Tx FIFO	8	W	数据传输寄存器

15.1.2 中断使能寄存器（IER）

中文名： 中断使能寄存器
寄存器位宽： [7：0]
偏移量： 0x01
复位值： 0x00

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	RW	保留
3	IME	1	RW	Modem 状态中断使能 ‘0’ – 关闭 ‘1’ – 打开
2	ILE	1	RW	接收器线路状态中断使能 ‘0’ – 关闭‘1’ – 打开
1	ITxE	1	RW	传输保存寄存器为空中断使能 ‘0’ – 关闭 ‘1’ – 打开
0	IRxE	1	RW	接收有效数据中断使能 ‘0’ – 关闭 ‘1’ – 打开

15.1.3 中断标识寄存器（IIR）

中文名： 中断源寄存器
寄存器位宽： [7：0]
偏移量： 0x02
复位值： 0xc1

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	R	保留
3:1	II	3	R	中断源表示位，详见下表
0	INTp	1	R	中断表示位

中断控制功能表

Bit 3	Bit 2	Bit 1	优先级	中断类型	中断源	中断复位控制
0	1	1	1st	接收线路状态	奇偶、溢出或帧错误，或打断中断	读 LSR

0	1	0	2nd	接收到有效数据	FIFO 的字符个数达到 trigger 的水平	FIFO 的字符个数低于 trigger 的值
1	1	0	2nd	接收超时	在 FIFO 至少有一个字符，但在 4 个字符时间内没有任何操作，包括读和写操作	读接收 FIFO
0	0	1	3rd	传输保存寄存器为空	传输保存寄存器为空	写数据到 THR 或者多 IIR
0	0	0	4th	Modem 状态	CTS, DSR, RI or DCD.	读 MSR

15.1.4 FIFO 控制寄存器（FCR）

中文名： FIFO 控制寄存器
寄存器位宽： [7：0]
偏移量： 0x02
复位值： 0xc0

位域	位域名称	位宽	访问	描述
7:6	TL	2	W	接收 FIFO 提出中断申请的 trigger 值 ‘00’ – 1 字节 ‘01’ – 4 字节 ‘10’ – 8 字节 ‘11’ – 14 字节
5:3	Reserved	3	W	保留
2	Txset	1	W	‘1’ 清除发送 FIFO 的内容，复位其逻辑
1	Rxset	1	W	‘1’ 清除接收 FIFO 的内容，复位其逻辑
0	Reserved	1	W	保留

15.1.5 线路控制寄存器（LCR）

中文名： 线路控制寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x03

位域	位域名称	位宽	访问	描述
7	dlab	1	RW	分频锁存器访问位 ‘1’ – 访问操作分频锁存器 ‘0’ – 访问操作正常寄存器
6	bcb	1	RW	打断控制位 ‘1’ – 此时串口的输出被置为 0(打断状态). ‘0’ – 正常操作
5	spb	1	RW	指定奇偶校验位 ‘0’ – 不用指定奇偶校验位 ‘1’ – 如果 LCR[4]位是 1 则传输和检查奇偶校验位为 0。如果 LCR[4]位是 0 则传输和检查奇偶校验位为 1。
4	eps	1	RW	奇偶校验位选择 ‘0’ – 在每个字符中有奇数个 1（包括数据和奇偶校验位） ‘1’ –在每个字符中有偶数个 1
3	pe	1	RW	奇偶校验位使能 ‘0’ – 没有奇偶校验位 ‘1’ – 在输出时生成奇偶校验位，输入则判断奇偶校验位

2	sb	1	RW	定义生成停止位的位数 '0' – 1 个停止位 '1' – 在 5 位字符长度时是 1.5 个停止位，其他长度是 2 个停止位
1:0	bec	2	RW	设定每个字符的位数 '00' – 5 位 '01' – 6 位 '10' – 7 位 '11' – 8 位

15.1.6 MODEM 控制寄存器（MCR）

中文名： Modem 控制寄存器
寄存器位宽： [7：0]
偏移量： 0x04
复位值： 0x00

位域	位域名称	位宽	访问	描述
7:5	Reserved	3	W	保留
4	Loop	1	W	回环模式控制位 '0' – 正常操作 '1' –回环模式。在在回环模式中，TXD 输出一直为 1，输出移位寄存器直接连到输入移位寄存器中。其他连接如下。 DTR ➔ DSR RTS ➔ CTS Out1 ➔ RI Out2 ➔ DCD

位域	位域名称	位宽	访问	描述
3	OUT2	1	W	在回环模式中连到 DCD 输入
2	OUT1	1	W	在回环模式中连到 RI 输入
1	RTSC	1	W	RTS 信号控制位
0	DTRC	1	W	DTR 信号控制位

15.1.7 线路状态寄存器（LSR）

中文名： 线路状态寄存器
寄存器位宽： [7：0]
偏移量： 0x05
复位值： 0x00

位域	位域名称	位宽	访问	描述
7	ERROR	1	R	错误表示位 ‘1’ – 至少有奇偶校验位错误，帧错误或打断中断的一个。 ‘0’ – 没有错误
6	TE	1	R	传输为空表示位 ‘1’ – 传输 FIFO 和传输移位寄存器都为空。给传输 FIFO 写数据时清零 ‘0’ – 有数据
5	TFE	1	R	传输 FIFO 位空表示位 ‘1’ – 当前传输 FIFO 为空，给传输 FIFO 写数据时清零 ‘0’ – 有数据

4	BI	1	R	打断中断表示位 ‘1’ –接收到 起始位＋数据＋奇偶位＋停止位都是 0，即有打断中断 ‘0’ – 没有打断
3	FE	1	R	帧错误表示位 ‘1’ – 接收的数据没有停止位 ‘0’ – 没有错误
2	PE	1	R	奇偶校验位错误表示位 ‘1’ – 当前接收数据有奇偶错误 ‘0’ – 没有奇偶错误
1	OE	1	R	数据溢出表示位 ‘1’ – 有数据溢出 ‘0’ – 无溢出
0	DR	1	R	接收数据有效表示位 ‘0’ – 在 FIFO 中无数据 ‘1’ – 在 FIFO 中有数据

对这个寄存器进行读操作时，LSR[4:1]和 LSR[7]被清零，LSR[6:5]在给传输 FIFO 写数据时清零，LSR[0]则对接收 FIFO 进行判断。

15.1.8 MODEM 状态寄存器 （MSR）

中文名： Modem 状态寄存器
寄存器位宽： [7：0]
偏移量： 0x06
复位值： 0x00

位域	位域名称	位宽	访问	描述
7	CDCD	1	R	DCD 输入值的反，或者在回环模式中连到 Out2
6	CRI	1	R	RI 输入值的反，或者在回环模式中连到 OUT1
5	CDSR	1	R	DSR 输入值的反，或者在回环模式中连到 DTR
4	CCTS	1	R	CTS 输入值的反，或者在回环模式中连到 RTS
3	DDCD	1	R	DDCD 指示位
2	TERI	1	R	RI 边沿检测。RI 状态从低到高变化
1	DDSR	1	R	DDSR 指示位
0	DCTS	1	R	DCTS 指示位

15.1.9 接收 FIFO 计数值 （RFC）

中文名：接收 FIFO 计数值

寄存器位宽： [7：0]

偏移量：0x08

复位值：0x00

位域	位域名称	位宽	访问	描述
7:0	RFC	8	R	反映当前接收 FIFO 中有效数据个数

15.1.10 发送 FIFO 计数值 （TFC）

中文名：发送 FIFO 计数值

寄存器位宽： [7：0]

偏移量：0x09

复位值：0x00

位域	位域名称	位宽	访问	描述
7:0	TFC	8	R	反映当前发送 FIFO 中有效数据个数

15.1.11 分频锁存器

中文名： 分频锁存器 1

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	LSB	8	RW	存放分频锁存器的低 8 位

中文名： 分频锁存器 2

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	MSB	8	RW	存放分频锁存器的高 8 位

中文名：分频锁存器 3

寄存器位宽： [7：0]

偏移量：0x02

复位值：0x00

位域	位域名称	位宽	访问	描述
7:0	D_DIV	8	RW	存放分频锁存器的小数分频值

15.1.12 新增寄存器的使用

新增的接收 FIFO 计数器（RFC）可供 CPU 检测接收 FIFO 中有效数据的个数，据此 CPU 可在收到一次中断后连续读取多个数据，提高 CPU 处理 UART 接收数据的能力；

发送 FIFO 计数器（TFC）可供 CPU 检测发送 FIFO 中有效数据的个数，据此 CPU 可在保证发送 FIFO 不溢出的前提下连续发送多个数据，提高 CPU 处理 UART 发送数据的能力；

分频锁存器 3（即小数分频寄存

器）用于解决仅用整数除法无法精确得到所需波特率的问题。以参考时钟 100MHz 除以 16，再除以波特率，所得商整数部分赋值给由分频器锁存器 MSB 和 LSB，小数部分乘以 256 赋值给分频锁存器 D_DIV。

15.2 SPI 控制器

SPI 控制器具有以下特性：

- 全双工同步串口数据传输
- 支持到 4 个的变长字节传输
- 主模式支持
- 模式故障产生错误标志并发出中断请求
- 双缓冲接收器
- 极性和相位可编程的串行时钟
- 可在等待模式下对 SPI 进行控制
- 支持从 SPI 启动
- 支持 Dual/Quad mode SPI flash

SPI 控制器寄存器物理地址基址为 0x1FE001F0。

表 15-1 SPI 控制器地址空间分布

地址名称	地址范围	大小
SPI Boot	0X1FC0_0000-0X1FD0_0000	1MByte
SPI Memory	0X1D00_0000-0X1E00_0000	16MByte
SPI Register	0X1FE0_01F0-0X1FE0_01FF	16Byte

SPI Boot 地址空间是系统启动时处理器最先访问的地址空间， 0xBFC00000 的地址被自动路由至 SPI。

SPI Memory 空间也可以通过 CPU 的读请求直接访问，其最低 1M 字节与 SPI BOOT 空间重叠。

15.2.1 控制寄存器（SPCR）

中文名： 控制寄存器
寄存器位宽： [7：0]
偏移量： 0x00
复位值： 0x10

位域	位域名称	位宽	访问	描述
7	Spie	1	RW	中断输出使能信号 高有效
6	spe	1	RW	系统工作使能信号高有效
5	Reserved	1	RW	保留
4	mstr	1	RW	master 模式选择位，此位一直保持 1
3	cpol	1	RW	时钟极性位
2	cpha	1	RW	时钟相位位 1 则相位相反，为 0 则相同
1:0	spr	2	RW	sclk_o 分频设定，需要与 sper 的 spre 一起使用

15.2.2 状态寄存器（SPSR）

中文名： 状态寄存器
寄存器位宽： [7：0]
偏移量： 0x01
复位值： 0x05

位域	位域名称	位宽	访问	描述
7	spif	1	RW	中断标志位 1 表示有中断申请，写 1 则清零
6	wcol	1	RW	写寄存器溢出标志位 为 1 表示已经溢出,写 1 则清零
5:4	Reserved	2	RW	保留
3	wffull	1	RW	写寄存器满标志 1 表示已经满
2	wfempty	1	RW	写寄存器空标志 1 表示空

1	rffull	1	RW	读寄存器满标志 1 表示已经满
0	rfempty	1	RW	读寄存器空标志 1 表示空

15.2.3 数据寄存器（TxFIFO）

中文名： 数据传输寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	Tx FIFO	8	W	数据传输寄存器

15.2.4 外部寄存器（SPER）

中文名： 外部寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:6	icnt	2	RW	在传输完多少个字节后送出中断申请信号 00 – 1 字节 01 - 2 字节 10 - 3 字节 11 - 3 字节
5:2	Reserved	4	RW	保留
1:0	spre	2	RW	与 Spr 一起设定分频的比率

分频系数：

spre	00	00	00	00	01	01	01	01	10	10	10	10
spr	00	01	10	11	00	01	10	11	00	01	10	11

分频系数	2	4	16	32	8	64	128	256	512	1024	2048	4096

15.2.5 参数控制寄存器（SFC_PARAM）

中文名： SPI Flash 参数控制寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x21

位域	位域名称	位宽	访问	描述
7:4	clk_div	4	RW	时钟分频数选择（分频系数与{spre,spr}组合相同）
3	dual_io	1	RW	使用双 I/O 模式，优先级高于快速读模式
2	fast_read	1	RW	使用快速读模式
1	burst_en	1	RW	spi flash 支持连续地址读模式
0	memory_en	1	RW	spi flash 读使能，无效时 csn[0]可由软件控制。

15.2.6 片选控制寄存器（SFC_SOFTCS）

中文名： SPI Flash 片选控制寄存器

寄存器位宽： [7：0]

偏移量： 0x05

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:4	csn	4	RW	csn 引脚输出值
3:0	csen	4	RW	为 1 时对应位的 cs 线由 7:4 位控制

15.2.7 时序控制寄存器（SFC_TIMING）

中文名： SPI Flash 时序控制寄存器

寄存器位宽： [7：0]

偏移量： 0x06

复位值： 0x03

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	RW	保留
3	quad_io	1	RW	4 线模式使能，1 有效

2	tFast	1	RW	
1:0	tCSH	2	RW	SPI Flash 的片选信号最短无效时间，以分频后时钟周期 T 计算 00: 1T 01: 2T 10: 4T 11: 8T

15.2.8 自定义控制寄存器（CTRL）

中文名： SPI Flash 自定义控制寄存器

寄存器位宽： [7：0]

偏移量： 0x08

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:4	nbyte	4	RW	一次传输的字节数
3:2	reserve	2	RW	保留
1	nbmode	1	RW	多字节传输模式
0	start	1	RW	开始多字节传输，完成后自动清零

15.2.9 自定义命令寄存器（CMD）

中文名： SPI Flash 自定义命令寄存器

寄存器位宽： [7：0]

偏移量： 0x09

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	cmd	8	RW	设置发送给 spi flash 的命令

15.2.10 自定义数据寄存器 0（BUF0）

中文名： SPI Flash 自定义数据寄存器 0

寄存器位宽： [7：0]

偏移量： 0x0a

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	buf0	8	RW	向 SPI 发送写命令时，该寄存器配置发送的第一个字节的数据；向 SPI 发送读命令时，该寄存器存储第一个读回来的数据。

15.2.11 自定义数据寄存器 1（BUF1）

中文名： SPI Flash 自定义数据寄存器 1

寄存器位宽： [7：0]

偏移量： 0x0b

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	buf1	8	RW	向 SPI 发送写命令时，该寄存器配置发送的第二个字节的数据；向 SPI 发送读命令时，该寄存器存储第二个读回来的数据。

15.2.12 自定义时序寄存器 0（TIMER0）

中文名： SPI Flash 自定义时序寄存器 0

寄存器位宽： [7：0]

偏移量： 0x0c

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	time0	8	RW	自定义命令所需时间值的低 8 位

15.2.13 自定义时序寄存器 1（TIMER1）

中文名： SPI Flash 自定义时序寄存器 1

寄存器位宽： [7：0]

偏移量： 0x0d

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	time1	8	RW	自定义命令所需时间值的中间 8 位

15.2.14 自定义时序寄存器 2（TIMER2）

中文名： SPI Flash 自定义时序寄存器 2

寄存器位宽： [7：0]

偏移量： 0x0e

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	time2	8	RW	自定义命令所需时间值的高 8 位

15.2.15 SPI 双线四线使用指南

除了传统的单线模式，SPI 控制器还支持以双线(dual mode)和四线（quad mode）两种工作模式从 SPI flash 启动。通过设置 dual_io 寄存器可以使 SPI 控制器进入双线模式，设置 quad_io 寄存器可以使 SPI 控制器进入四线模式。可以在 BIOS 代码的前几条指令中增加对这两个寄存器的配置代码，配置完成后控制器即按照配置对应的工作模式进行取指，以此可提高开机速度。

需要注意的是，有的 SPI FLASH 默认并没有使能四线模式，或者在四线模式下需要配置时序相关的参数(如 dummy clocks)。为了增加 SPI 控制器对各种 FLASH 的适用性，本控制器增加自定义的寄存器(0x8-0xe)。其具体使用方法为：

1. 设置自定义命令寄存器（CMD）(0x9)，该寄存器为向 SPI FLASH 发送的命令；
2. 如果 SPI FLASH 要求本次发送的命令需要过一段时间才完成，则把等待的时间配置到自定义时序寄存器 TIMER0-TIMER2(0xc-0xe)中，否则这些寄存器保持默认值 0；
3. 如果向 SPI FLASH 写配置信息，则需要把配置信息写入自定义数据寄存器 BUF0-BUF1(0xa-0xb)；如果向 SPI FLASH 读配置信息，则这两个寄存器存储读回来的值；
4. 配置自定义控制寄存器 CTRL[7：1]其中 CTRL[1](nbmode)代表将进行多字节传输模

式，此次传输字节数通过 CTRL[7:4](nbyte)给定；

5. 配置自定义控制寄存器 CTRL[0]开始此次传输。

一般来说，所需要配置的寄存器位于 FLASH 的非易失性存储区，所以以上配置仅需要配置一次。

15.3 I2C 控制器

本章给出 I2C 的详细描述和配置使用。本系统芯片集成了 I2C 接口，主要用于实现两个器件之间数据的交换。I2C 总线是由数据线 SDA 和时钟 SCL 构成的串行总线，可发送和接收数据。器件与器件之间进行双向传送，最高传送速率 400kbps。

龙芯 3A4000 中集成的 I2C 控制器既可以作为主设备，也可以作为从设备，这两种模式之间通过配置内部寄存器进行切换。作为从设备时，仅用于读取芯片内部温度，从设备的地址由寄存器 SLV_CTRL[6:0]指定。

I2C0 控制器寄存器物理地址基址为 0x1FE00120。

I2C1 控制器寄存器物理地址基址为 0x1FE00130。

下面对具体的内部寄存器进行说明。

15.3.1 分频锁存器低字节寄存器（PRERlo）

中文名：分频锁存器低字节寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0xff

位域	位域名称	位宽	访问	描述
7:0	PRERlo	8	RW	存放分频锁存器的低 8 位

15.3.2 分频锁存器高字节寄存器（PRERhi）

中文名：分频锁存器高字节寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0xff

位域	位域名称	位宽	访问	描述
7:0	PRERhi	8	RW	存放分频锁存器的高 8 位

假设分频锁存器的值为 prescale，从 LPB 总线 PCLK 时钟输入的频率为 clock_a，SCL 总线的输出频率为 clock_s，则应满足如下关系：

```
Prcescale  =  clock_a/(4*clock_s)-1
```

15.3.3 控制寄存器（CTR）

中文名：控制寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x20

位域	位域名称	位宽	访问	描述
7	EN	1	RW	模块工作使能位 为 1 正常工作模式, 0 对分频寄存器进行操作
6	IEN	1	RW	中断使能位为 1 则打开中断
5	MST_EN	1	RW	模块主从选择 0：slave 模式 1：master 模式
4:0	Reserved	5	RW	保留

15.3.4 发送数据寄存器（TXR）

中文名：发送寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:1	DATA	7	W	存放下个将要发送的字节
0	DRW	1	W	当数据传送时，该位保存的是数据的最低位； 当地址传送时，该位指示读写状态

15.3.5 接收数据寄存器（RXR）

中文名：接收寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	RXR	8	R	存放最后一个接收到的字节

15.3.6 命令控制寄存器（CR）

中文名：命令寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x00

位域	位域名称	位宽	访问	描述
7	STA	1	W	产生 START 信号
6	STO	1	W	产生 STOP 信号
5	RD	1	W	产生读信号
4	WR	1	W	产生写信号
3	ACK	1	W	产生应答信号
2:1	Reserved	2	W	保留
0	IACK	1	W	产生中断应答信号

都是在 I2C 发送数据后硬件自动清零。对这些位读操作时候总是读回‘0’。bit 3 为 1 时表示此次传输结束时控制器不发送 ack，反之结束时发送 ack。

15.3.7 状态寄存器（SR）

中文名：状态寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x00

位域	位域名称	位宽	访问	描述
7	RxACK	1	R	收到应答位 1 没收到应答位 0 收到应答位
6	Busy	1	R	I2c 总线忙标志位 1 总线在忙

				0 总线空闲
5	AL	1	R	当 I2C 核失去 I2C 总线控制权时，该位置 1
4:2	Reserved	3	R	保留
1	TIP	1	R	指示传输的过程 1 表示正在传输数据 0 表示数据传输完毕
0	IF	1	R	中断标志位，一个数据传输完，或另外一个器件发起数据传输，该位置 1

15.3.8 从设备控制寄存器（SLV_CTRL）

中文名：从设备控制寄存器

寄存器位宽： [7：0]

偏移量： 0x07

复位值： 0x00

位域	位域名称	位宽	访问	描述
7	SLV_EN	1	WR	从模式使能，当 MST_EN 为 0 时起作用，可用于复位从机内部逻辑
6：0	SLV_ADDR	7	WR	从模式 I2C 地址，可通过软件配置

16 3A3000 内核兼容性

为了实现从 3A3000 开始 Linux 内核的向后兼容，必须对现有的内核根据芯片的实现规范进行一些改造。

而为了实现兼容 3A3000 的内核，3A4000 芯片除了实现一套按新规范的配置方法之外，也需要对目前内核中广泛使用的机制进行支持。

以下从内核兼容和新特性支持两个方面对 3A4000 的内核进行介绍。

16.1 兼容 3A3000 内核

为了对 3A3000 的内核进行兼容，必须对内核中的以下部分进行修改。

16.1.1 处理器特性识别方法

对于 MIPS 处理器，内核中并没有使用一种通用的办法对处理器的不同特性进行识别，而是通过 PRID 对处理器型号进行区分，再根据处理器型号在不同的场合下进行不同的处理。因为目前内核里只针对已有的处理器型号进行了判断并处理，对还没有实现的新处理器并没有默认的处理方法，导致运行在新处理器上时很多底层代码没有对应的实现。

为了解决这一问题，从 3A4000 开始，实现一套处理器配置指令，以及处理器特性识别的指令，来规范软硬件接口。可以通过处理器配置指令进行访问，也可以通过 0x3ff00000 的基地址进行访问。寄存器记为 CSR[偏移地址][位]。

该寄存器标识了一些软件相关的处理器特性，供软件在使能特定功能前查看。寄存器的偏移地址 0x0008。记为 CSR[0x08]。

表 16-1 芯片特性寄存器

位域	字段名	访问	复位值	描述
0	Centigrade	R	1'b1	CSR[0x428]有效
1	Node counter	R	1'b1	CSR[0x408]有效
2	MSI	R	1'b1	MSI 可用
3	EXT_IOI	R	1'b1	EXT_IOI 可用
4	IPI_percore	R	1'b1	通过 CSR 私有地址进行 IPI 发送

5	Freq_percore	R	1'b1	通过 CSR 私有地址调整频率
6	Freq_scale	R	1'b0	动态分频功能可用
7	DVFS_v1	R	1'b0	动态调频 v1 可用
8	Tsensor	R	1'b0	温度传感器可用

16.1.2 当前内核改动方法

当前 3.10 内核中，有六处使用 PRID 进行功能特性识别的代码，为了对未来的处理器进行支持，需要对其中的五处进行修改。

这五个函数如下：

函数	路径	描述
cpu_probe_loongson	arch/mips/kernel/cpu-probe.c	进行芯片型号的识别
loonson_cpu_temp	driver/platform/mips/cpu_hwmon.c	读取片上温度传感器
play_dead	arch/mips/loongson/loongson-3/smp.c	动态开关核支持
init_node_counter_clocksource	arch/mips/loongson/loongson-3/node_counter.c	使能片上时钟源
ls7a_init_irq	arch/mips/loongson/loongson-3/ls7a-irq.c	使能 MSI 中断

（1） cpu_probe_loongson

该函数用于芯片型号的识别，需要在原有的 default 条件中增加使用处理器配置指令识别厂商名称、芯片名称并给对应数据结构赋值的代码。对应的寄存器如下。

厂商名称寄存器。CSR[0x0010]。

表 16-2 厂商名称寄存器

位域	字段名	访问	复位值	描述
63:0	Vendor	R	0x6e6f7367_6e6f6f4c	字符串“Loongson”

芯片名称寄存器。CSR[0x0020]。

表 16-3 芯片名称寄存器

位域	字段名	访问	复位值	描述
63:0	ID	R	0x00003030_30344133	字符串“3A4000”

（2） loongson_cpu_temp

该函数用于读取片上温度传感器，需要在原有的 default 条件中增加新的处理。首先根据 CSR[0x8][0]判断是否有片上温度传感器，决定是否使用处理器配置指令读取片上温度寄存器 CSR[0x428]。

（3） play_dead

该函数用于动态开关核，为了未来的处理器，需要对这个函数进行较大的改动。原有的函数根据 PRID 调用不同的函数进行针对性的刷 Cache 操作并进行关核，需要改为根据 MIPS 手册第二卷的规范，从相关的 CP0 寄存器中，读出的 ICACHE/DCACHE/VCACHE 的路数、大小的配置，进行对应的刷 Cache 操作，再进行关核。要注意，在所有的龙芯 2 号和龙芯 3 号系列处理器中，通过 CP0 寄存器读取 cache 配置信息时，CP0.config2 中的 Secondary Cache 指的是片上末级 Cache 即 scache；当存在片内私有二级 Cache 时，通过 CP0.config2 中的 Tertiary Cache 表示。关核时，需要根据 CSR[0x420][23]是否为 1 决定是使用 CSR[0x1050][3]进行关核还是使用 0x3ff001d0 的对应位进行关核。

（4） init_node_counter_clocksource

该函数用于初始化片上时钟源，需要增加一个 default 条件，根据 CSR[0x8][1]，确定是否有 node_counter。同时还需要增加一个参数，不再对某些特定值进行修正。

（5） ls7a_init_irq

该函数用于决定是否使用 MSI 中断，需要增加一个 default 条件，在各种已知处理器之外，根据 CSR[0x8][2]，确定是否有 MSI 支持。

16.2 新特性支持

为了在内核中使用 3A4000 处理器提供的新特性，可以根据以下的方法进行识别或使能。在这里只对能够提升系统性能的部分进行介绍，而诸如通过 CSR 指令发送核间中断的新机制，因为兼容 3A3000 处理器的要求，实际上还必须采用现有的特定地址访问方式，没有必要特意进行 CSR 的支持，反而增加软件开销。

16.2.1 处理器特性识别

新特性都是通过 CSR 寄存器指令进行识别，为了支持新特性，还需要考虑不支持处理器配置指令的旧处理器的处理方法。一是在各个需要进行特性识别的位置都增加 PRID 的判断，将已有的 PRID 都进行处理，二是增加异常指令处理，当保留指令被识别为处理器配置指令

时，根据指令内容，根据 PRID，构建正确的返回值。

16.2.2 扩展中断模式

为了在内核中使能扩展中断模式，需要按以下的顺序进行设置。

1） 扩展中断模式支持通过 CSR[0x8][3]进行识别。

2） PMON 中需要将期望支持扩展中断模式的 HT 控制器的外部中断转换寄存器配置为正确的值。其寄存器定义如下，设置为如下值：

INT_trans_en = 0//使用 CSR 寄存器进行使能控制，CSR[0x420][48]与该寄存器都可使能扩展中断模式，在 PMON 中默认不使能该模式，由内核配置 CSR[0x420][48]打开

INT_trans_allow = 1//允许外部使能中断转换功能

INT_trans_addr = 0x1000000001140//扩展中断寄存器地址，见 14.3.3。

INT_trans_cache = 0//Uncache 方式

偏移量： 0x270

复位值： 0x00000000

名称： HT RX INT TRANS Lo

表 16-4 HT RX INT TRANS LO

位域	位域名称	位宽	复位值	访问	描述
31:4	INT_trans_addr[31:4]	28	0x0	R/W	中断转换地址低位
3:0	Reserved	4	0x0	R	保留

偏移量： 0x274

复位值： 0x00000000

名称： HT RX INT TRANS Hi

表 16-5 HT RX INT TRANS Hi

位域	位域名称	位宽	复位值	访问	描述
31	INT_trans_en	1	0x0	R/W	中断转换使能
30	INT_trans_allow	1	0x0	R/W	中断转换允许
29:26	INT_trans_cache	4	0x0	R/W	中断转换 Cache 域
25:0	INT_trans_addr[58:32]	26	0x0	R/W	中断转换地址高位

3） 内核先通过 CSR[0x8][3]识别出扩展中断模式支持，再通过寄存器 CSR[0x420][48]使能扩展中断模式。基地址为 0x1fe00000，偏移地址 0x0420。

表 16-6 其它功能设置寄存器

位域	字段名	访问	复位值	描述
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能

4） 设置扩展中断模式的相应路由及内部控制。

16.3 配置寄存器指令调试支持

配置寄存器指令原则上在使用时不跨片访问，但为了满足对调试等功能的需求，在此使用多个寄存器地址对跨片访问进行支持。值得注意的是，这类寄存器只能写，不能读。

加上原有的核间中断等可以跨片访问的寄存器，所有的这类寄存器及地址如下。

表 16-7 处理器核核间通信寄存器

名称	偏移地址	权限	描述
IPI_Send	0x1040	WO	32 位中断分发寄存器 [31] 等待完成标志，置 1 时会等待中断生效 [30:26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 中断向量号，对应 IPI_Status 中的向量
Mail_Send	0x1048	WO	64 位 MailBox 缓存寄存器 [63:32] MailBox 数据 [31] 等待完成标志，置 1 时会等待写入生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:2] MailBox 号 0 - MailBox0 低 32 位 1 - MailBox0 高 32 位 2 - MailBox1 低 32 位 3 - MailBox1 高 32 位 4 - MailBox2 低 32 位 5 - MailBox2 高 32 位

			6 - MailBox3 低 32 位 7 - MailBox4 高 32 位 [1:0] 保留
FREQ_Send	0x1058	WO	32 位频率使能寄存器 [31] 等待完成标志，置 1 时会等待中断生效 [30:26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 写入对应的处理器核私有频率配置寄存器。CSR[0x1050]
ANY_Send	0x1158	WO	64 位寄存器访问寄存器 [63:32] 写入数据 [31] 等待完成标志，置 1 时会等待中断生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 目标处理器核号 [15:0] 写入的寄存器偏移地址